Can Large Language Models Understand Symbolic Graphics Programs?

Zeju Qiu[1,†] **Weiyang Liu**[1,2,†,*] **Haiwen Feng**[1,†] **Zhen Liu**[1,‡] **Tim Z. Xiao**[1,‡] **Katherine M. Collins**[2,‡]
Joshua B. Tenenbaum[3] **Adrian Weller**[2] **Michael J. Black**[1] **Bernhard Schölkopf**[1]
[1]Max Planck Institute for Intelligent Systems, Tübingen [2]University of Cambridge [3]MIT
[†]Joint first author [‡]Joint second author [*]Project lead **sgp-bench.github.io**

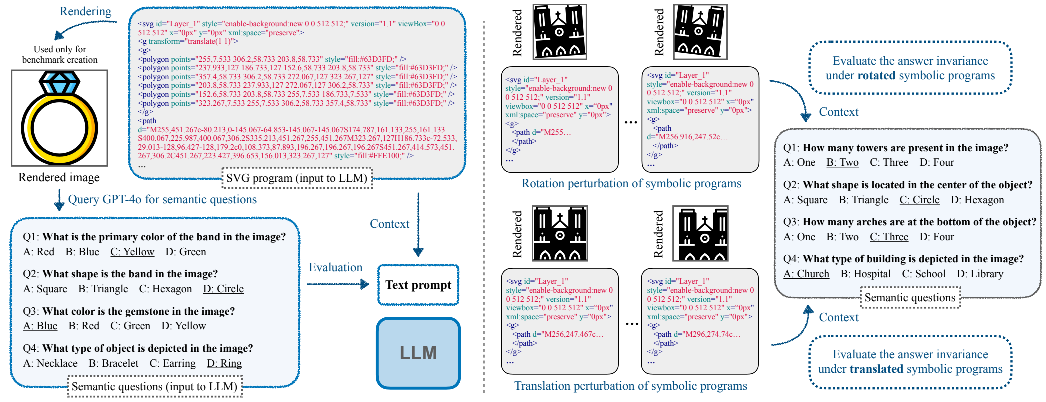
Abstract

Against the backdrop of enthusiasm for large language models (LLMs), there is a growing need to scientifically assess their capabilities and shortcomings. This is nontrivial in part because it is difficult to find tasks which the models have not encountered during training. Utilizing symbolic graphics programs, we propose a domain well-suited to test multiple spatial-semantic reasoning skills of LLMs. Popular in computer graphics, these programs procedurally generate visual data. While LLMs exhibit impressive skills in general program synthesis and analysis, symbolic graphics programs offer a new layer of evaluation: they allow us to test an LLM's ability to answer semantic questions about the images or 3D geometries without a vision encoder. To semantically understand the symbolic programs, LLMs would need to possess the ability to "imagine" and reason how the corresponding graphics content would look with only the symbolic description of the local curvatures and strokes. We use this task to evaluate LLMs by creating a large benchmark for the semantic visual understanding of symbolic graphics programs, built procedurally with minimal human effort. Particular emphasis is placed on transformations of images that leave the image level semantics invariant while introducing significant changes to the underlying program. We evaluate commercial and open-source LLMs on our benchmark to assess their ability to reason about visual output of programs, finding that LLMs considered stronger at reasoning generally perform better. Lastly, we introduce a novel method to improve this ability – *Symbolic Instruction Tuning* (SIT), in which the LLM is finetuned with pre-collected instruction data on symbolic graphics programs. Interestingly, we find that SIT not only improves LLM's understanding on symbolic programs, but it also improves general reasoning ability on various other benchmarks.

1 Introduction

What are large language models (LLMs) capable of? Recent studies [5, 58] have shown that LLMs are able to generate generic computer programs, indicating a degree of pragmatic understanding of the symbolic structure of programs. Motivated by this progress, we focus on another important family of computer programs, called symbolic graphics programs, where a graphics content (*e.g.*, image, 3D asset) can be generated by running a program. We are interested in the following question: *Can large language models "understand" symbolic graphics programs?*

Before trying to answer this question, we start by defining what we consider "understanding" of symbolic graphics programs, in the context of this work. Because a (deterministic) graphics program can be uniquely rendered to an image (the graphics programs we consider here), we characterize LLMs' understanding of the graphics program as the semantic understanding of the corresponding rendered image. More specifically, we approximate such a semantic visual understanding by the ability to correctly answer semantic questions only based on the raw program input. These semantic questions are generated based on the rendered image, such that they are easy to answer given the image and yet challenging given only the program as text prompt. Guided by this insight, we propose a generic pipeline for creating benchmarks that can evaluate this particular ability of LLMs to understand symbolic graphics programs, while requiring minimal human effort. While we of course recognize that there are other elements of visual reasoning that characterize understanding in humans, and that we ought to evaluate in machine intelligence, we believe that our benchmark provides insight into one element of "understanding" of symbolic graphics programs that helps assay what current LLMs are (and are not) capable of.


Rendering
Used only for
benchmark creation
Rendered image
SVG program (input to LLM)
Query GPT-4o for semantic questions
Context
Evaluation
Text prompt
LLM
Semantic questions (input to LLM)
Rendered
Rotation perturbation of symbolic programs
Translation perturbation of symbolic programs
Evaluate the answer invariance
under rotated symbolic programs
Evaluate the answer invariance
under translated symbolic programs
Context
Semantic questions

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0
512 512" x="0px" y="0px" xml:space="preserve">
<g transform="translate(1 1)">
<g>
<polygon points="255,7.533 306.2,58.733 203.8,58.733" style="fill:#63D3FD;" />
<polygon points="237.933,127 186.733,127 152.6,58.733 203.8,58.733" style="fill:#63D3FD;" />
<polygon points="357.4,58.733 306.2,58.733 272.067,127 323.267,127" style="fill:#63D3FD;" />
<polygon points="203.8,58.733 237.933,127 272.067,127 306.2,58.733" style="fill:#63D3FD;" />
<polygon points="152.6,58.733 203.8,58.733 255,7.533 186.733,7.533" style="fill:#63D3FD;" />
<polygon points="323.267,7.533 255,7.533 306.2,58.733 357.4,58.733" style="fill:#63D3FD;" />
</g>
<path
d="M255,451.267c-80.213,0-145.067-64.853-145.067-145.067S174.787,161.133,255,161.133
S400.067,225.987,400.067,306.2S335.213,451.267,255,451.267M323.267,127H186.733c-72.533,
29.013-128,96.427-128,179.2c0,108.373,87.893,196.267,196.267,196.267S451.267,414.573,451.
267,306.2C451.267,223.427,396.653,156.013,323.267,127" style="fill:#FFE100;" />
...
```

Q1: **What is the primary color of the band in the image?**
A: Red B: Blue C: Yellow D: Green

Q2: **What shape is the band in the image?**
A: Square B: Triangle C: Hexagon D: Circle

Q3: **What color is the gemstone in the image?**
A: Blue B: Red C: Green D: Yellow

Q4: **What type of object is depicted in the image?**
A: Necklace B: Bracelet C: Earring D: Ring

```
<svg id="Layer_1"
style="enable-background:new 0
0 512 512;" version="1.1"
viewbox="0 0 512 512" x="0px"
xml:space="preserve" y="0px">
<g>
  <path d="M255…
  </path>
</g>
...
```

```
<svg id="Layer_1"
style="enable-background:new 0
0 512 512;" version="1.1"
viewbox="0 0 512 512" x="0px"
xml:space="preserve" y="0px">
<g>
  <path
d="M256.916,247.52c…
  </path>
</g>
...
```

```
<svg id="Layer_1"
style="enable-background:new 0
0 512 512;" version="1.1"
viewbox="0 0 512 512" x="0px"
xml:space="preserve" y="0px">
<g>
  <path d="M256,247.467c…
  </path>
</g>
...
```

```
<svg id="Layer_1"
style="enable-background:new 0
0 512 512;" version="1.1"
viewbox="0 0 512 512" x="0px"
xml:space="preserve" y="0px">
<g>
  <path d="M296,274.74c…
  </path>
</g>
...
```

Q1: **How many towers are present in the image?**
A: One B: Two C: Three D: Four

Q2: **What shape is located in the center of the object?**
A: Square B: Triangle C: Circle D: Hexagon

Q3: **How many arches are at the bottom of the object?**
A: One B: Two C: Three D: Four

Q4: **What type of building is depicted in the image?**
A: Church B: Hospital C: School D: Library

(a) Semantic Understanding of Symbolic Graphics Programs

(b) Consistency of Symbolic Graphics Program Understanding

Figure 1: Our benchmark assesses LLMs' understanding of symbolic graphics programs in semantic understanding and prediction consistency. Note that the LLM can only see symbolic graphics programs and the corresponding questions. The rendered images are not input to the LLM.

The semantic understanding of symbolic graphics programs is particularly interesting in the following aspects. First, such a semantic understanding may necessitate a form of "visual imagination" ability that enables the LLM to "imagine" how the produced graphics content visually looks like. If a LLM can perfectly answer every possible semantic question of a symbolic graphics program (with sufficient knowledge, the graphics content can be reconstructed), then we can say that such a LLM may have a good "visual imagination" of this program. Second, symbolic programs represent a procedural way to generate the graphics content, and hence the semantic understanding also requires the long-range sequential reasoning of the program. The order of the symbolic operations may substantially affect its semantic meaning, making the problem quite challenging. Third, many semantic questions involve an accurate grounding of semantic components, and such a grounding in the symbolic program requires a fine-grained understanding of the program structure. This motivates us to study whether LLMs have the ability to semantically understand symbolic graphics programs, and furthermore, how to improve this ability. In general, correctly answering semantic questions about symbolic graphics programs requires a combination of multiple sophisticated reasoning abilities from LLMs, which therefore makes the task of symbolic graphics program understanding an ideal benchmark to contribute towards evaluating the holistic reasoning capabilities of LLMs. Reasoning over symbolic programs is of particular interest from a cognitive perspective as well [103, 84, 25]. To what extent LLMs can operate over such a representation with rich structures remains an open problem.

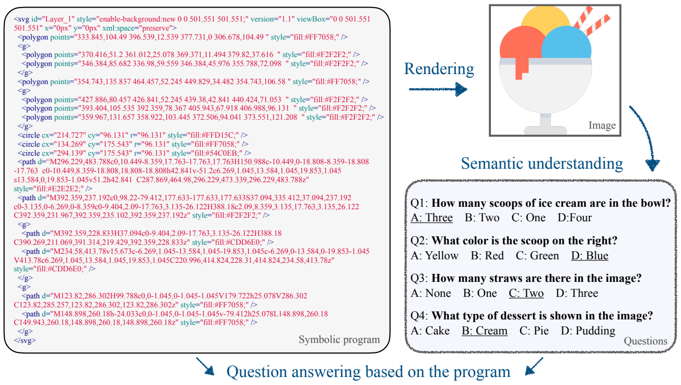
Motivated by the significance of symbolic graphics program understanding, we build a benchmark, called *SGP-Bench*, for two important variants of symbolic graphics programs: scalable vector graphics (SVG) as a generic language for representing 2D vector graphics, and customized computer-aided design (CAD) as a domain-specific language (DSL) for representing 2D/3D objects. Our benchmark consists of two types of evaluations. (1) *Semantic understanding*: We construct a number of semantic questions (*i.e.*, multiple-choice questions with 4 options) from a set of images (from multiple different categories). These questions are fed to LLMs along with the symbolic program to evaluate the semantic understanding. (2) *Semantic consistency*: To evaluate the robustness of LLM's semantic understanding, we perform random translation and rotation to the original symbolic programs and then test the same semantic questions based on the perturbed programs. We evaluate the consistency of the answers from LLMs using these perturbed symbolic programs with identical semantic meaning. This evaluation can also help lower the possibility of test data leakage, because the randomly perturbed programs are unlikely to be seen during pretraining. An overview of SGP-Bench is given in Figure 1. We further validate our automated labels via a human study (see Appendix B).

In addition to performing evaluation under the common in-context setting wherein LLMs are used "out-of-the-box" and not finetuned, we also evaluate whether finetuning LLMs on a curated dataset can boost performance. To this end, we propose *Symbolic Instruction Tuning* (SIT). The key idea is to collect an instruction dataset based on the rendered images. Because the semantic questions of interest are usually easy to answer from the visual input, we take advantage of the rendered images (that correspond to symbolic programs) and query a powerful language-vision model (*e.g.*, GPT-4o) for detailed captioning. This leads to a scalable way to collect an instruction dataset for symbolic graphics programs. Then, we simply finetune open-source LLMs on this dataset. Our experiments demonstrate that SIT can improve a model's semantic understanding of symbolic programs, and more importantly, its general reasoning ability. Our contributions are summarized below:

- We introduce a new task of symbolic graphics program understanding and propose a generic yet highly scalable benchmark creation pipeline for this task.
- We build a large benchmark, SGP-Bench, for comprehensively evaluating LLM's semantic understanding and consistency of symbolic graphics programs. In SGP-Bench, we consider two types of symbolic graphics programs: SVG for 2D vector graphics and CAD for 2D/3D objects.
- To improve the symbolic program understanding, we collect an instruction-following dataset and propose a new finetuning method, called symbolic instruction tuning.
- We introduce a symbolic MNIST dataset, where the symbolic program understanding can be challenging so that even a powerful LLM (*i.e.*, GPT-4o) can only obtain nearly a chance-level performance while the rendered image remains very easy to recognize.
- Finally, we find that SIT can also improve general reasoning performance.

2 Semantic Understanding of Symbolic Graphics Programs

We introduce the task of semantic symbolic graphics program understanding. Our goal is to assess to what extent a LLM is able to "understand" a symbolic graphics program, which may begin to belie some latent capability to "visually imagine". Specifically, we leverage the correspondence between deterministic symbolic graphics programs and rendered images, and then characterize the understanding of symbolic graphics programs as the semantic understanding of the corresponding rendered image. To do so, we use the performance on question-answering to evaluate the semantic understanding of images. The same set of questions, along with the corresponding symbolic graphics programs, are then used to evaluate the symbolic program understanding of LLMs (the rendered image will not be used here). Figure 2 gives an illustration of symbolic graphics program understanding. The intuition behind this evaluation is that, if an LLM has a good sense of the symbolic graphics and implicit de-rendering, then the LLM should have a rough understanding about its rendered image such that it is able to answer arbitrary semantic questions about the rendered image.



```
<svg id="Layer_1" style="enable-background:new 0 0 501.551 501.551;" version="1.1" viewBox="0 0 501.551
501.551" x="0px" y="0px" xml:space="preserve">
 <polygon points="333.845,104.49 396.539,12.539 377.731,0 306.678,104.49 " style="fill:#FF7058;" />
 <g>
  <polygon points="370.416,51.2 361.012,25.078 369.371,11.494 379.82,37.616  " style="fill:#F2F2F2;" />
  <polygon points="346.384,85.682 336.98,59.559 346.384,45.976 355.788,72.098  " style="fill:#F2F2F2;" />
 </g>
 <polygon points="354.743,135.837 464.457,52.245 449.829,34.482 354.743,106.58 " style="fill:#FF7058;" />
 <g>
  <polygon points="427.886,80.457 426.841,52.245 439.38,42.841 440.424,71.053  " style="fill:#F2F2F2;" />
  <polygon points="393.404,105.535 392.359,78.367 405.943,67.918 406.988,96.131  " style="fill:#F2F2F2;" />
  <polygon points="359.967,131.657 358.922,103.445 372.506,94.041 373.551,121.208  " style="fill:#F2F2F2;" />
 </g>
 <circle cx="214.727" cy="96.131" r="96.131" style="fill:#FFD15C;" />
 <circle cx="134.269" cy="175.543" r="96.131" style="fill:#FF7058;" />
 <circle cx="294.139" cy="175.543" r="96.131" style="fill:#54C0EB;" />
 <path d="M296.229,483.788c0,10.449-8.359,17.763-17.763,17.763H150.988c-10.449,0-18.808-8.359-18.808
-17.763  c0-10.449,8.359-18.808,18.808-18.808h42.841v-51.2c6.269,1.045,13.584,1.045,19.853,1.045
s13.584,0,19.853-1.045v51.2h42.841  C287.869,464.98,296.229,473.339,296.229,483.788z"
style="fill:#E2E2E2;" />
 <path d="M392.359,237.192c0,98.22-79.412,177.633-177.633,177.633S37.094,335.412,37.094,237.192
c0-3.135,0-6.269,0-8.359c0-9.404,2.09-17.763,3.135-26.122H388.18c2.09,8.359,3.135,17.763,3.135,26.122
C392.359,231.967,392.359,235.102,392.359,237.192z" style="fill:#F2F2F2;" />
 <g>
  <path d="M392.359,228.833H37.094c0-9.404,2.09-17.763,3.135-26.122H388.18
C390.269,211.069,391.314,219.429,392.359,228.833z" style="fill:#CDD6E0;" />
  <path d="M234.58,413.78v15.673c-6.269,1.045-13.584,1.045-19.853,1.045c-6.269,0-13.584,0-19.853-1.045
V413.78c6.269,1.045,13.584,1.045,19.853,1.045C220.996,414.824,228.31,414.824,234.58,413.78z"
style="fill:#CDD6E0;" />
 </g>
 <g>
  <path d="M123.82,286.302H99.788c0,0-1.045,0-1.045-1.045V179.722h25.078V286.302
C123.82,285.257,123.82,286.302,123.82,286.302z" style="fill:#FF7058;" />
  <path d="M148.898,260.18h-24.033c0,0-1.045,0-1.045-1.045v-79.412h25.078L148.898,260.18
C149.943,260.18,148.898,260.18,148.898,260.18z" style="fill:#FF7058;" />
 </g>
</svg>
```

Q1: How many scoops of ice cream are in the bowl?
A: Three B: Two C: One D:Four

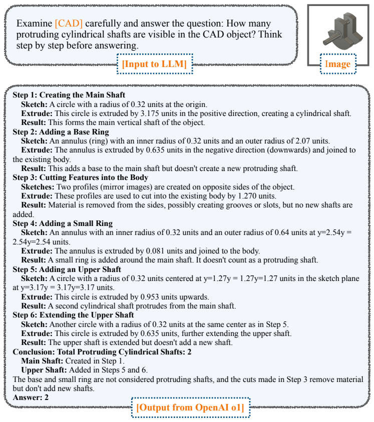
Q2: What color is the scoop on the right?
A: Yellow B: Red C: Green D: Blue

Q3: How many straws are there in the image?
A: None B: One C: Two D: Three

Q4: What type of dessert is shown in the image?
A: Cake B: Cream C: Pie D: Pudding

Figure 2: Illustration of the symbolic graphics program understanding task.

Symbolic graphics program understanding can be viewed as a form of visual question answering in the sense that visual input is represented by a symbolic program representation. Compared to current vision-language models [56, 55, 121] that encodes images with a text-aligned encoder [76], we consider the case where the visual input are encoded by a symbolic program that can exactly recover the graphics content. From this perspective, our task aims to uncover the potential of using symbolic programs as a representation to perform visual reasoning.



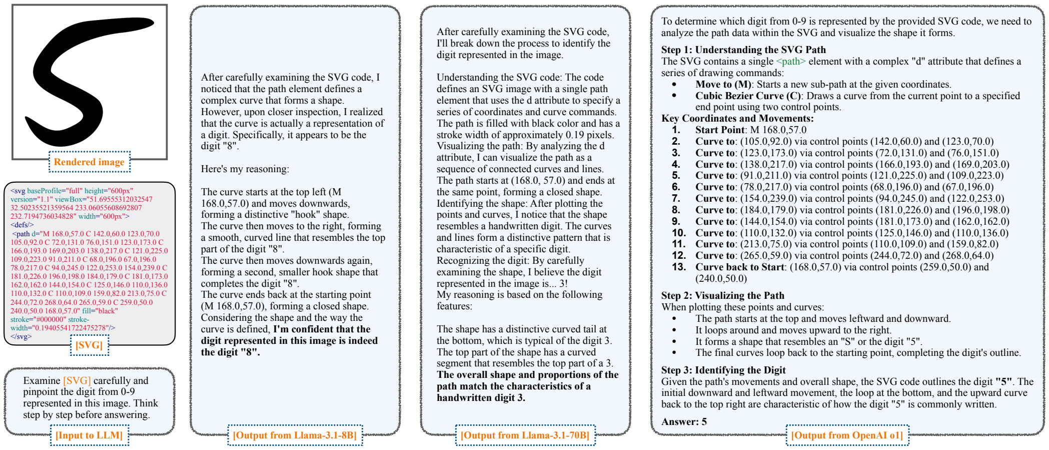
Examine [CAD] carefully and answer the question: How many protruding cylindrical shafts are visible in the CAD object? Think step by step before answering.

Step 1: Creating the Main Shaft
Sketch: A circle with a radius of 0.32 units at the origin.
Extrude: This circle is extruded by 3.175 units in the positive direction, creating a cylindrical shaft.
Result: This forms the main vertical shaft of the object.
Step 2: Adding a Base Ring
Sketch: An annulus (ring) with an inner radius of 0.32 units and an outer radius of 2.07 units.
Extrude: The annulus is extruded by 0.635 units in the negative direction (downwards) and joined to the existing body.
Result: This adds a base to the main shaft but doesn't create a new protruding shaft.
Step 3: Cutting Features into the Body
Sketches: Two profiles (mirror images) are created on opposite sides of the object.
Extrude: These profiles are used to cut into the existing body by 1.270 units.
Result: Material is removed from the sides, possibly creating grooves or slots, but no new shafts are added.
Step 4: Adding a Small Ring
Sketch: An annulus with an inner radius of 0.32 units and an outer radius of 0.64 units at y=2.54y = 2.54y=2.54 units.
Extrude: The annulus is extruded by 0.081 units and joined to the body.
Result: A small ring is added around the main shaft. It doesn't count as a protruding shaft.
Step 5: Adding an Upper Shaft
Sketch: A circle with a radius of 0.32 units centered at y=1.27y = 1.27y=1.27 units in the sketch plane at y=3.17y = 3.17y=3.17 units.
Extrude: This circle is extruded by 0.953 units upwards.
Result: A second cylindrical shaft protrudes from the main shaft.
Step 6: Extending the Upper Shaft
Sketch: Another circle with a radius of 0.32 units at the same center as in Step 5.
Extrude: This circle is extruded by 0.635 units, further extending the upper shaft.
Result: The upper shaft is extended but doesn't add a new shaft.
Conclusion: Total Protruding Cylindrical Shafts: 2
Main Shaft: Created in Step 1.
Upper Shaft: Added in Steps 5 and 6.
The base and small ring are not considered protruding shafts, and the cuts made in Step 3 remove material but don't add new shafts.
Answer: 2

Figure 3: A qualitative example of CAD reasoning.

3 Why is Understanding Symbolic Graphics Programs Interesting?

To showcase why semantic understanding of symbolic graphics problems require multiple sophisticated reasoning abilities, we provide a few qualitative examples of LLM's output in Figure 3 and Figure 4. Figure 3 shows a qualitative example of how OpenAI-o1 reasons over the CAD program. The reasoning process is highly nontrivial, as it requires multiple reasoning abilities, such as numeric perception, spatial reasoning, geometric understanding, long-range planning and common sense. This is also part of



```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="51.6955531203254 7
32.50235521359564 233.06055608692807
232.7194736034828" width="600px">
<defs/>
 <path d="M 168.0,57.0 C 142.0,60.0 123.0,70.0
105.0,92.0 C 72.0,131.0 76.0,151.0 123.0,173.0 C
166.0,193.0 169.0,203.0 138.0,217.0 C 121.0,225.0
109.0,223.0 91.0,211.0 C 68.0,196.0 67.0,196.0
78.0,217.0 C 94.0,245.0 122.0,253.0 154.0,239.0 C
181.0,226.0 196.0,198.0 184.0,179.0 C 181.0,173.0
162.0,162.0 144.0,154.0 C 125.0,146.0 110.0,136.0
110.0,132.0 C 110.0,109.0 159.0,82.0 213.0,75.0 C
244.0,72.0 268.0,64.0 265.0,59.0 C 259.0,50.0
240.0,50.0 168.0,57.0" fill="black"
stroke="#000000" stroke-
width="0.19405541722475278"/>
</svg>
```

Examine [SVG] carefully and pinpoint the digit from 0-9 represented in this image. Think step by step before answering.

After carefully examining the SVG code, I noticed that the path element defines a complex curve that forms a shape. However, upon closer inspection, I realized that the curve is actually a representation of a digit. Specifically, it appears to be the digit "8".

Here's my reasoning:

The curve starts at the top left (M 168.0,57.0) and moves downwards, forming a distinctive "hook" shape.
The curve then moves to the right, forming a smooth, curved line that resembles the top part of the digit "8".
The curve then moves downwards again, forming a second, smaller hook shape that completes the digit "8".
The curve ends back at the starting point (M 168.0,57.0), forming a closed shape.
Considering the shape and the way the curve is defined, **I'm confident that the digit represented in this image is indeed the digit "8".**

After carefully examining the SVG code, I'll break down the process to identify the digit represented in the image.

Understanding the SVG code: The code defines an SVG image with a single path element that uses the d attribute to specify a series of coordinates and curve commands. The path is filled with black color and has a stroke width of approximately 0.19 pixels.
Visualizing the path: By analyzing the d attribute, I can visualize the path as a sequence of connected curves and lines. The path starts at (168.0, 57.0) and ends at the same point, forming a closed shape.
Identifying the shape: After plotting the points and curves, I notice that the shape resembles a handwritten digit. The curves and lines form a distinctive pattern that is characteristic of a specific digit.
Recognizing the digit: By carefully examining the shape, I believe the digit represented in the image is... 3!
My reasoning is based on the following features:

The shape has a distinctive curved tail at the bottom, which is typical of the digit 3.
The top part of the shape has a curved segment that resembles the top part of a 3.
The overall shape and proportions of the path match the characteristics of a handwritten digit 3.

To determine which digit from 0-9 is represented by the provided SVG code, we need to analyze the path data within the SVG and visualize the shape it forms.

Step 1: Understanding the SVG Path
The SVG contains a single <path> element with a complex "d" attribute that defines a series of drawing commands:

- **Move to (M)**: Starts a new sub-path at the given coordinates.
- **Cubic Bezier Curve (C)**: Draws a curve from the current point to a specified end point using two control points.

Key Coordinates and Movements:

1. **Start Point**: M 168.0,57.0
2. **Curve to**: (105.0,92.0) via control points (142.0,60.0) and (123.0,70.0)
3. **Curve to**: (123.0,173.0) via control points (72.0,131.0) and (76.0,151.0)
4. **Curve to**: (138.0,217.0) via control points (166.0,193.0) and (169.0,203.0)
5. **Curve to**: (91.0,211.0) via control points (121.0,225.0) and (109.0,223.0)
6. **Curve to**: (78.0,217.0) via control points (68.0,196.0) and (67.0,196.0)
7. **Curve to**: (154.0,239.0) via control points (94.0,245.0) and (122.0,253.0)
8. **Curve to**: (184.0,179.0) via control points (181.0,226.0) and (196.0,198.0)
9. **Curve to**: (144.0,154.0) via control points (181.0,173.0) and (162.0,162.0)
10. **Curve to**: (110.0,132.0) via control points (125.0,146.0) and (110.0,136.0)
11. **Curve to**: (213.0,75.0) via control points (110.0,109.0) and (159.0,82.0)
12. **Curve to**: (265.0,59.0) via control points (244.0,72.0) and (268.0,64.0)
13. **Curve back to Start**: (168.0,57.0) via control points (259.0,50.0) and (240.0,50.0)

Step 2: Visualizing the Path
When plotting these points and curves:

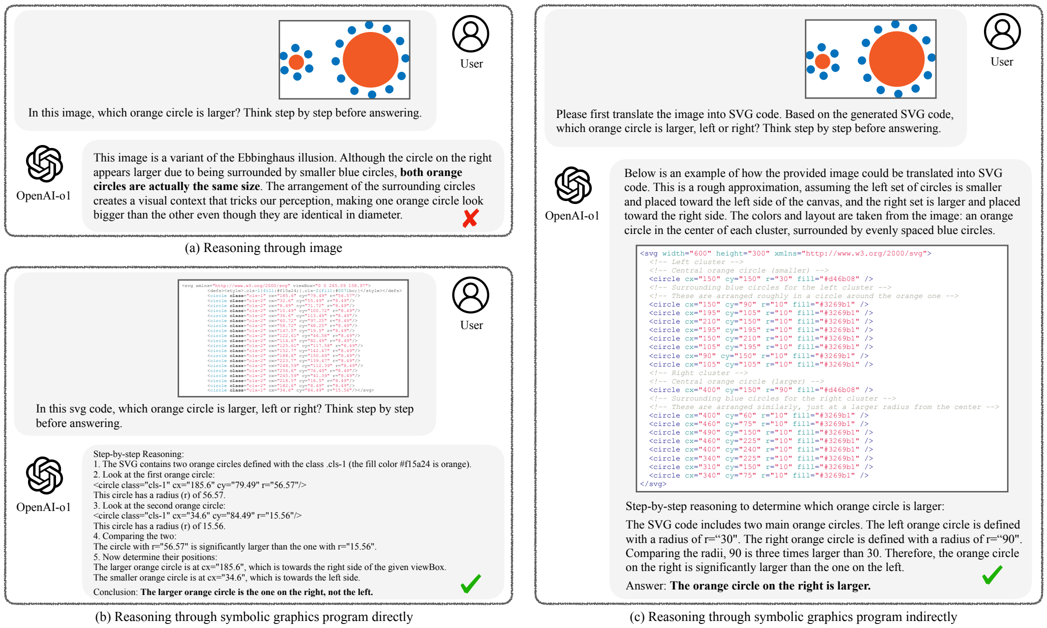
- The path starts at the top and moves leftward and downward.
- It loops around and moves upward to the right.
- It forms a shape that resembles an "S" or the digit "5".
- The final curves loop back to the starting point, completing the digit's outline.

Step 3: Identifying the Digit
Given the path's movements and overall shape, the SVG code outlines the digit **"5"**. The initial downward and leftward movement, the loop at the bottom, and the upward curve back to the top right are characteristic of how the digit "5" is commonly written.

Answer: 5

Figure 4: Qualitative examples of how LLMs reason over the symbolic program and obtain their answers.

the reason that SGP-Bench can well evaluate the general reasoning ability of LLMs. In Figure 4, we query different LLMs (from weak to strong: Llama-3.1-8B, Llama-3.1-70B, OpenAI-o1) by asking which digit this given SVG program represents. We can observe that all LLMs start to reason from the low-level curves and gradually build up its understanding from local components to semantic elements. Specifically, we can observe that LLMs understand the symbolic graphics program through line-by-line reasoning and then combine the detailed reasoning into an overall semantic understanding. This process is particularly intriguing because it shows that LLMs understand symbolic programs through actual reasoning rather than memorization. More interestingly, the more powerful LLMs, *e.g.*, OpenAI-o1, show better general understanding of the symbolic graphics program and fine-grained grounding of the program, which is consistent with the results on SGP-Bench.



In this image, which orange circle is larger? Think step by step before answering.

This image is a variant of the Ebbinghaus illusion. Although the circle on the right appears larger due to being surrounded by smaller blue circles, **both orange circles are actually the same size**. The arrangement of the surrounding circles creates a visual context that tricks our perception, making one orange circle look bigger than the other even though they are identical in diameter.

(a) Reasoning through image

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 265.09 158.97">
        <defs><style>.cls-1{fill:#f15a24;}.cls-2{fill:#0071bc;}</style></defs>
        <circle class="cls-1" cx="185.6" cy="79.49" r="56.57"/>
        <circle class="cls-2" cx="32.6" cy="55.49" r="8.49"/>
        <circle class="cls-2" cx="8.49" cy="71.72" r="8.49"/>
        <circle class="cls-2" cx="10.49" cy="100.72" r="8.49"/>
        <circle class="cls-2" cx="36.6" cy="113.49" r="8.49"/>
        <circle class="cls-2" cx="60.72" cy="97.25" r="8.49"/>
        <circle class="cls-2" cx="58.72" cy="68.25" r="8.49"/>
        <circle class="cls-2" cx="147.5" cy="19.5" r="8.49"/>
        <circle class="cls-2" cx="122.61" cy="46.58" r="8.49"/>
        <circle class="cls-2" cx="114.6" cy="82.49" r="8.49"/>
        <circle class="cls-2" cx="125.61" cy="117.58" r="8.49"/>
        <circle class="cls-2" cx="152.7" cy="142.47" r="8.49"/>
        <circle class="cls-2" cx="188.6" cy="150.49" r="8.49"/>
        <circle class="cls-2" cx="223.7" cy="139.47" r="8.49"/>
        <circle class="cls-2" cx="248.59" cy="112.39" r="8.49"/>
        <circle class="cls-2" cx="256.6" cy="76.49" r="8.49"/>
        <circle class="cls-2" cx="245.59" cy="41.39" r="8.49"/>
        <circle class="cls-2" cx="218.5" cy="16.5" r="8.49"/>
        <circle class="cls-2" cx="182.6" cy="8.49" r="8.49"/>
        <circle class="cls-1" cx="34.6" cy="84.49" r="15.56"/></svg>
```

In this svg code, which orange circle is larger, left or right? Think step by step before answering.

Step-by-step Reasoning:
1. The SVG contains two orange circles defined with the class .cls-1 (the fill color #f15a24 is orange).
2. Look at the first orange circle:
<circle class="cls-1" cx="185.6" cy="79.49" r="56.57"/>
This circle has a radius (r) of 56.57.
3. Look at the second orange circle:
<circle class="cls-1" cx="34.6" cy="84.49" r="15.56"/>
This circle has a radius (r) of 15.56.
4. Comparing the two:
The circle with r="56.57" is significantly larger than the one with r="15.56".
5. Now determine their positions:
The larger orange circle is at cx="185.6", which is towards the right side of the given viewBox.
The smaller orange circle is at cx="34.6", which is towards the left side.

Conclusion: **The larger orange circle is the one on the right, not the left.**

(b) Reasoning through symbolic graphics program directly

Please first translate the image into SVG code. Based on the generated SVG code, which orange circle is larger, left or right? Think step by step before answering.

Below is an example of how the provided image could be translated into SVG code. This is a rough approximation, assuming the left set of circles is smaller and placed toward the left side of the canvas, and the right set is larger and placed toward the right side. The colors and layout are taken from the image: an orange circle in the center of each cluster, surrounded by evenly spaced blue circles.

```
<svg width="600" height="300" xmlns="http://www.w3.org/2000/svg">
  <!-- Left cluster -->
  <!-- Central orange circle (smaller) -->
  <circle cx="150" cy="150" r="30" fill="#d46b08" />
  <!-- Surrounding blue circles for the left cluster -->
  <!-- These are arranged roughly in a circle around the orange one -->
  <circle cx="150" cy="90" r="10" fill="#3269b1" />
  <circle cx="195" cy="105" r="10" fill="#3269b1" />
  <circle cx="210" cy="150" r="10" fill="#3269b1" />
  <circle cx="195" cy="195" r="10" fill="#3269b1" />
  <circle cx="150" cy="210" r="10" fill="#3269b1" />
  <circle cx="105" cy="195" r="10" fill="#3269b1" />
  <circle cx="90" cy="150" r="10" fill="#3269b1" />
  <circle cx="105" cy="105" r="10" fill="#3269b1" />
  <!-- Right cluster -->
  <!-- Central orange circle (larger) -->
  <circle cx="400" cy="150" r="90" fill="#d46b08" />
  <!-- Surrounding blue circles for the right cluster -->
  <!-- These are arranged similarly, just at a larger radius from the center -->
  <circle cx="400" cy="60" r="10" fill="#3269b1" />
  <circle cx="460" cy="75" r="10" fill="#3269b1" />
  <circle cx="490" cy="150" r="10" fill="#3269b1" />
  <circle cx="460" cy="225" r="10" fill="#3269b1" />
  <circle cx="400" cy="240" r="10" fill="#3269b1" />
  <circle cx="340" cy="225" r="10" fill="#3269b1" />
  <circle cx="310" cy="150" r="10" fill="#3269b1" />
  <circle cx="340" cy="75" r="10" fill="#3269b1" />
</svg>
```

Step-by-step reasoning to determine which orange circle is larger:

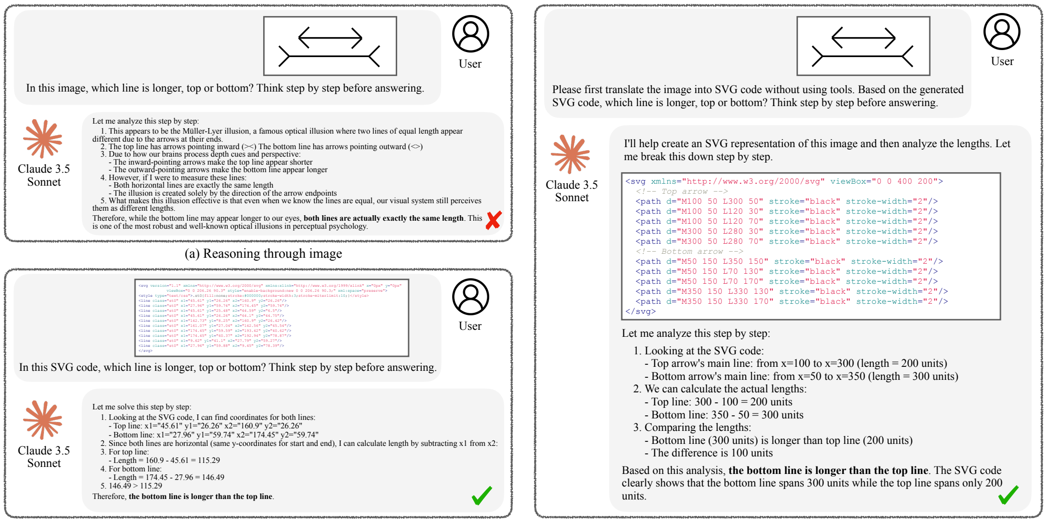
The SVG code includes two main orange circles. The left orange circle is defined with a radius of r=“30". The right orange circle is defined with a radius of r=“90". Comparing the radii, 90 is three times larger than 30. Therefore, the orange circle on the right is significantly larger than the one on the left.

Answer: **The orange circle on the right is larger.**

(c) Reasoning through symbolic graphics program indirectly

Figure 5: OpenAI-o1 still suffers from the spurious correlation from the Ebbinghaus illusion while reasoning over images (a). In contrast, OpenAI-o1 works perfectly fine while reasoning over symbolic graphics programs directly (b) or indirectly (c).

To highlight the importance of symbolic graphics programs, we give another example in Figure 5, where a powerful LLM like OpenAI-o1 can suffer from the visual spurious correlation, and in contrast, this spurious correlation can be avoided when using the symbolic graphics program as the visual representation. Specifically, we construct a visual example that resembles the Ebbinghaus illusion, but the conclusion differs from the classic Ebbinghaus illusion (*i.e.*, two orange circles are of the same size, although they look different) as we intentionally make one of the orange circle obviously larger than the other (shown in Figure 5). Then we feed this curated example to OpenAI-o1 and ask which orange circle is larger. We compare three cases: (a) image input; (b) symbolic program input;



In this image, which line is longer, top or bottom? Think step by step before answering.

Let me analyze this step by step:

1. This appears to be the Müller-Lyer illusion, a famous optical illusion where two lines of equal length appear different due to the arrows at their ends.
2. The top line has arrows pointing inward (><) The bottom line has arrows pointing outward (<>)
3. Due to how our brains process depth cues and perspective:
   - The inward-pointing arrows make the top line appear shorter
   - The outward-pointing arrows make the bottom line appear longer
4. However, if I were to measure these lines:
   - Both horizontal lines are exactly the same length
   - The illusion is created solely by the direction of the arrow endpoints
5. What makes this illusion effective is that even when we know the lines are equal, our visual system still perceives them as different lengths.

Therefore, while the bottom line may appear longer to our eyes, **both lines are actually exactly the same length**. This is one of the most robust and well-known optical illusions in perceptual psychology. ✗

(a) Reasoning through image

```
<svg version="1.1" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
         viewBox="0 0 206.26 90.3" style="enable-background:new 0 0 206.26 90.3;" xml:space="preserve">
<style type="text/css">.st0{fill:none;stroke:#000000;stroke-width:3;stroke-miterlimit:10;}</style>
<line class="st0" x1="45.61" y1="26.26" x2="160.9" y2="26.26"/>
<line class="st0" x1="27.96" y1="59.74" x2="174.45" y2="59.74"/>
<line class="st0" x1="45.61" y1="25.48" x2="64.59" y2="6.5"/>
<line class="st0" x1="45.61" y1="26.26" x2="64.1" y2="44.75"/>
<line class="st0" x1="142.73" y1="8.25" x2="160.9" y2="26.42"/>
<line class="st0" x1="161.07" y1="27.04" x2="142.56" y2="45.54"/>
<line class="st0" x1="174.45" y1="59.59" x2="193.42" y2="40.62"/>
<line class="st0" x1="174.45" y1="60.37" x2="192.94" y2="78.87"/>
<line class="st0" x1="9.62" y1="41.1" x2="27.79" y2="59.27"/>
<line class="st0" x1="27.96" y1="59.88" x2="9.45" y2="78.39"/>
</svg>
```

In this SVG code, which line is longer, top or bottom? Think step by step before answering.

Let me solve this step by step:

1. Looking at the SVG code, I can find coordinates for both lines:
   - Top line: x1="45.61" y1="26.26" x2="160.9" y2="26.26"
   - Bottom line: x1="27.96" y1="59.74" x2="174.45" y2="59.74"
2. Since both lines are horizontal (same y-coordinates for start and end), I can calculate length by subtracting x1 from x2:
3. For top line:
   - Length = 160.9 - 45.61 = 115.29
4. For bottom line:
   - Length = 174.45 - 27.96 = 146.49
5. 146.49 > 115.29

Therefore, **the bottom line is longer than the top line**. ✓

(b) Reasoning through symbolic graphics program directly

Please first translate the image into SVG code without using tools. Based on the generated SVG code, which line is longer, top or bottom? Think step by step before answering.

I'll help create an SVG representation of this image and then analyze the lengths. Let me break this down step by step.

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 400 200">
  <!-- Top arrow -->
  <path d="M100 50 L300 50" stroke="black" stroke-width="2"/>
  <path d="M100 50 L120 30" stroke="black" stroke-width="2"/>
  <path d="M100 50 L120 70" stroke="black" stroke-width="2"/>
  <path d="M300 50 L280 30" stroke="black" stroke-width="2"/>
  <path d="M300 50 L280 70" stroke="black" stroke-width="2"/>
  <!-- Bottom arrow -->
  <path d="M50 150 L350 150" stroke="black" stroke-width="2"/>
  <path d="M50 150 L70 130" stroke="black" stroke-width="2"/>
  <path d="M50 150 L70 170" stroke="black" stroke-width="2"/>
  <path d="M350 150 L330 130" stroke="black" stroke-width="2"/>
  <path d="M350 150 L330 170" stroke="black" stroke-width="2"/>
</svg>
```

Let me analyze this step by step:

1. Looking at the SVG code:
   - Top arrow's main line: from x=100 to x=300 (length = 200 units)
   - Bottom arrow's main line: from x=50 to x=350 (length = 300 units)
2. We can calculate the actual lengths:
   - Top line: 300 - 100 = 200 units
   - Bottom line: 350 - 50 = 300 units
3. Comparing the lengths:
   - Bottom line (300 units) is longer than top line (200 units)
   - The difference is 100 units

Based on this analysis, **the bottom line is longer than the top line**. The SVG code clearly shows that the bottom line spans 300 units while the top line spans only 200 units. ✓

(c) Reasoning through symbolic graphics program indirectly

Figure 6: Claude-3.5-Sonnet also suffers from the spurious correlation from the Müller-Lyer illusion.

and (c) image input with indirect symbolic program. As long as we first ask the LLM to translate the image into symbolic program, the LLM will no longer suffer from the spurious correlation. The same phenomenon also appears for the Müller-Lyer illusion using Claude-3.5-Sonnet (Figure 6). These examples validate the advantages of reasoning the visual world through symbolic graphics programs.

4 A Benchmark for Symbolic Graphics Program Understanding

4.1 Dataset Creation Pipeline

To construct our benchmark, we need questions about a symbolic program, based on its rendered image. To build a large benchmark, it is essential to consider how we can scale up the question collection effectively, with minimal human effort. To this end, we use a powerful vision-language model (*e.g.*, GPT-4o) to generate semantic questions based on the rendered images, and then we inspect them manually to make sure that these questions are reasonable and the answer to them is correct. We also run a human study over a randomized set of 500 of the automatically generated questions along with the corresponding images, and find high agreement (see Appendix B). The overall procedure for our dataset creation is given in Figure 7. In this pipeline, the rendering of symbolic programs and the GPT-4o querying are both scalable and can be done with minimal human involvement. Human annotators then inspect the generated question-answer pairs based on the rendered image, which requires much less efforts than manually writing questions and answers. We emphasize that this program-question-answer triplet creation method is general, as it works for most of the symbolic graphics programs. SVG and 2D CAD programs can directly produce 2D images, so it is straightforward to use this pipeline. For 3D CAD programs, they produce 3D models and we first render them into 2D images with a few fixed camera positions. These rendered images are used to query GPT-4o, and the following procedures are identical to the SVG case.

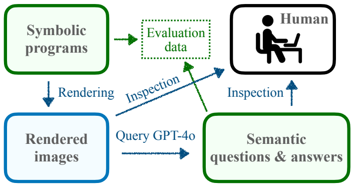
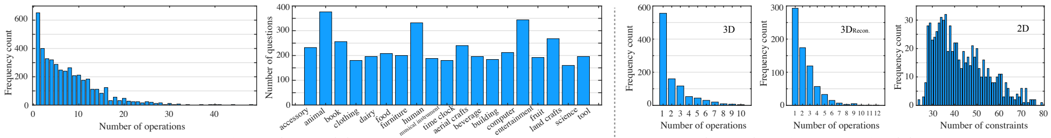


Figure 7: Dataset construction procedure.



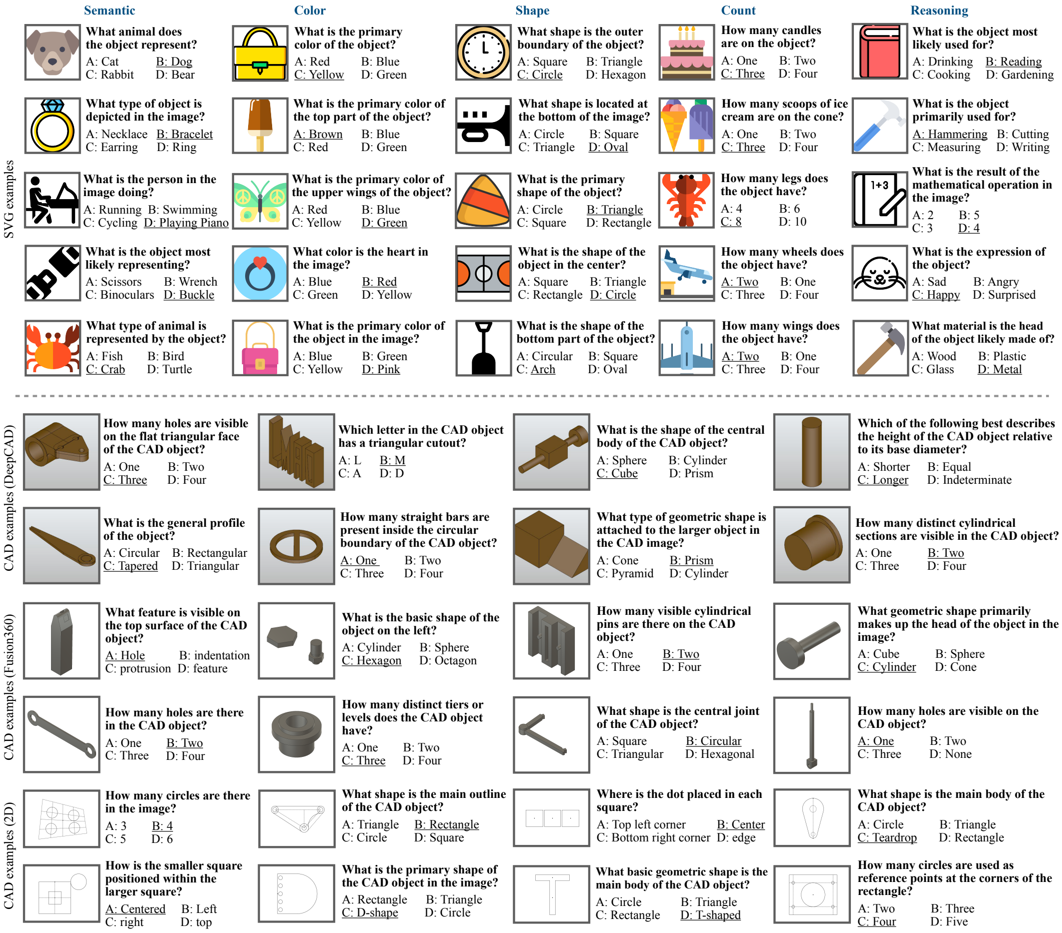
(a) SVG dataset statistics summary

(b) CAD dataset (3D, $3D_{Recon}$, 2D) statistics summary

Figure 8: Key dataset statistics for both SVG and CAD programs. We show the distribution of the operation number in a program for both SVG and CAD data, together with the number of examples of each category in the SVG dataset.

4.2 Benchmarking Semantic Understanding

SVG dataset statistics. We collect in total 1,085 SVG programs covering 19 categories, and each program has 4 semantic multiple-choice questions (with 4 options), resulting in a total of



What animal does the object represent?
A: Cat B: Dog
C: Rabbit D: Bear

What is the primary color of the object?
A: Red B: Blue
C: Yellow D: Green

What shape is the outer boundary of the object?
A: Square B: Triangle
C: Circle D: Hexagon

How many candles are on the object?
A: One B: Two
C: Three D: Four

What is the object most likely used for?
A: Drinking B: Reading
C: Cooking D: Gardening

What type of object is depicted in the image?
A: Necklace B: Bracelet
C: Earring D: Ring

What is the primary color of the top part of the object?
A: Brown B: Blue
C: Red D: Green

What shape is located at the bottom of the image?
A: Circle B: Square
C: Triangle D: Oval

How many scoops of ice cream are on the cone?
A: One B: Two
C: Three D: Four

What is the object primarily used for?
A: Hammering B: Cutting
C: Measuring D: Writing

What is the person in the image doing?
A: Running B: Swimming
C: Cycling D: Playing Piano

What is the primary color of the upper wings of the object?
A: Red B: Blue
C: Yellow D: Green

What is the primary shape of the object?
A: Circle B: Triangle
C: Square D: Rectangle

How many legs does the object have?
A: 4 B: 6
C: 8 D: 10

What is the result of the mathematical operation in the image?
A: 2 B: 5
C: 3 D: 4

What is the object most likely representing?
A: Scissors B: Wrench
C: Binoculars D: Buckle

What color is the heart in the image?
A: Blue B: Red
C: Green D: Yellow

What is the shape of the object in the center?
A: Square B: Triangle
C: Rectangle D: Circle

How many wheels does the object have?
A: Two B: One
C: Three D: Four

What is the expression of the object?
A: Sad B: Angry
C: Happy D: Surprised

What type of animal is represented by the object?
A: Fish B: Bird
C: Crab D: Turtle

What is the primary color of the object in the image?
A: Blue B: Green
C: Yellow D: Pink

What is the shape of the bottom part of the object?
A: Circular B: Square
C: Arch D: Oval

How many wings does the object have?
A: Two B: One
C: Three D: Four

What material is the head of the object likely made of?
A: Wood B: Plastic
C: Glass D: Metal

How many holes are visible on the flat triangular face of the CAD object?
A: One B: Two
C: Three D: Four

Which letter in the CAD object has a triangular cutout?
A: L B: M
C: A D: D

What is the shape of the central body of the CAD object?
A: Sphere B: Cylinder
C: Cube D: Prism

Which of the following best describes the height of the CAD object relative to its base diameter?
A: Shorter B: Equal
C: Longer D: Indeterminate

What is the general profile of the object?
A: Circular B: Rectangular
C: Tapered D: Triangular

How many straight bars are present inside the circular boundary of the CAD object?
A: One B: Two
C: Three D: Four

What type of geometric shape is attached to the larger object in the CAD image?
A: Cone B: Prism
C: Pyramid D: Cylinder

How many distinct cylindrical sections are visible in the CAD object?
A: One B: Two
C: Three D: Four

What feature is visible on the top surface of the CAD object?
A: Hole B: indentation
C: protrusion D: feature

What is the basic shape of the object on the left?
A: Cylinder B: Sphere
C: Hexagon D: Octagon

How many visible cylindrical pins are there on the CAD object?
A: One B: Two
C: Three D: Four

What geometric shape primarily makes up the head of the object in the image?
A: Cube B: Sphere
C: Cylinder D: Cone

How many holes are there in the CAD object?
A: One B: Two
C: Three D: Four

How many distinct tiers or levels does the CAD object have?
A: One B: Two
C: Three D: Four

What shape is the central joint of the CAD object?
A: Square B: Circular
C: Triangular D: Hexagonal

How many holes are visible on the CAD object?
A: One B: Two
C: Three D: None

How many circles are there in the image?
A: 3 B: 4
C: 5 D: 6

What shape is the main outline of the CAD object?
A: Triangle B: Rectangle
C: Circle D: Square

Where is the dot placed in each square?
A: Top left corner B: Center
C: Bottom right corner D: edge

What shape is the main body of the CAD object?
A: Circle B: Triangle
C: Teardrop D: Rectangle

How is the smaller square positioned within the larger square?
A: Centered B: Left
C: right D: top

What is the primary shape of the CAD object in the image?
A: Rectangle B: Triangle
C: D-shape D: Circle

What basic geometric shape is the main body of the CAD object?
A: Circle B: Triangle
C: Rectangle D: T-shaped

How many circles are used as reference points at the corners of the rectangle?
A: Two B: Three
C: Four D: Five

Figure 9: Example questions for SVG and CAD programs. Due to the space limit, we omit the programs and only show the rendered images.

4, 340 questions. We make sure that answers are evenly distributed across 4 options. Detailed dataset statistics are in Figure 8(a). Our SVG benchmark consists of 5 different types of questions, including "Semantic": 1, 085 questions, "Color": 864 questions, "Shape": 1, 217 questions, "Count": 819 questions, and "Reasoning": 355 questions. "Semantic" tests the global semantic meaning of the object represented by SVG codes, while the other four question types focus on detailed, local understanding of the object. "Color" is color-related questions about specific object parts, which evaluates the localization of the corresponding semantic part. "Count" is about counting the occurrences of certain patterns or semantic parts. "Shape" is about the shape of certain parts of the object, either finding the exact definition of the geometric shape or finding geometric shapes that resemble the object part. Figure 9 gives some SVG examples.

CAD dataset statistics. We collect 2, 400 CAD programs from three different datasets [101, 105, 86]. The CAD dataset consists of 1000 programs from DeepCAD [105], which forms the *3D* subset; 700 programs from the Fusion360 Reconstruction Dataset [101], which constitutes the $3D_{complex}$ subset; and 700 programs from SketchGraphs [86], which makes up the *2D* subset (as shown in Table 1). Different from SVG, there is no generally established syntax for building CAD models from graphics codes; each of our 3 CAD subsets follows a different language syntax, with varying levels of complexity. When benchmarking LLMs for CAD tasks, we include domain-specific language syntax rules as part of the input prompt. The LLM is required to apply in-context learning of this syntax and answer the test questions. Then we feed the renderings to GPT-4o and generate one semantic multiple-choice question (with 4 options) and its answer. This gives us 2, 400 questions in total. We make sure that ground truth answers are evenly distributed across 4 options. Detailed dataset statistics are given in Figure 8(b). Some examples from our CAD dataset are provided in Figure 9.

Experimental results and discussion. We find that graphics program understanding, as we operationalize it here, is challenging. The average accuracy of all models (proprietary and open-source) is below 70% (ranging from 30% to 67%) on SVG and below 75% (ranging from 28% to 74%) on

CAD. Among these, SVG makes it more difficult for the models to understand as these 2D graphics contain richer semantics. Significant performance improvements are observed **in line with scaling laws** [113], as larger model sizes consistently lead to gains across various open-source LLMs. For example, Llama-3's score is improved from 0.429 to 0.548 on SVG and from 0.633 to 0.694 when its size increased from 8B to 70B, Qwen-1.5's from 0.376 to 0.499 on SVG and 0.486 to 0.632 on CAD with the size from 7B to 110B. We also notice consistent improvements given the same model size and model family but from different generations. For example, Qwen-72B from 0.466 to 0.537, Llama-8B from 0.429 to 0.465 and Llama-70B from 0.548 to 0.574. The consistent performance gain on both the SVG and CAD indicates that the semantic understanding of symbolic graphics programs, including both 2D and 3D understanding, is a fundamental capability that is aligned with the scaling law of LLMs, even if these models have not been specifically trained for this task.

Compared to the open-sourced LLMs that we consider here, proprietary models (GPTs) and (Claudes) outperform most of them by a large margin. Within the family of current most popular GPTs, we see a performance improvement with a 27% boost (from GPT-3.5's 0.498 to GPT-4's 0.633) when evaluating these GPT variants on our SGP-Bench. This result is aligned with the seeming improvement of reasoning ability in the GPT family, validating that SGP-Bench can well distinguish different LLMs. The overall best-performing model in both SVG and CAD is Claude 3.5 Sonnet.

The semantic understanding of graphics programs can be probed across different aspects, ranging from attribute-level investigations of "color" and "shape" to higher-level discussions of "semantics", "counting" and "reasoning". Our benchmark is designed to cover these investigations for LLMs. Most LLMs perform well on color-related questions, followed by shape-related questions, with "count" and "semantic" questions showing progressively lower performance. This consistency is intriguing, as it resembles the coarse-to-fine structure of visual information processing. "Color" is the most visually salient feature, "shape" understanding requires a finer grasp of global and local structures, and "count" and "semantic" questions demand deeper comprehension and knowledge. The difficulty curve is evident, with most open-source models achieving roughly half the accuracy on semantic questions compared to color questions. For instance, the best-performing open-source model, Llama3.1-405B, achieves 37.6% accuracy on semantics and 81.6% accuracy on color grounding tasks. While open-source models struggle with "semantic" questions, ChatGPT performs quite well, with semantics being their second-best category after color grounding.

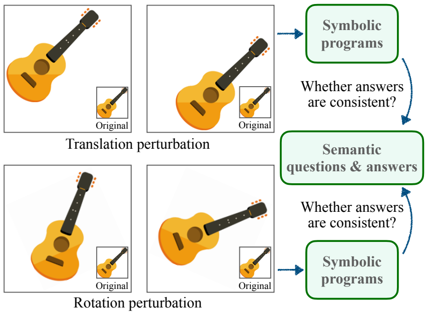


Figure 10: The semantic consistency test assesses if semantic understanding remains the same when the program is perturbed without semantically changing its rendered content. Image perturbations result in significant code-level changes because symbolic programs rely on absolute coordinates.

4.3 Benchmarking Semantic Consistency

Large language models (LLMs) are exposed to vast amounts of online data, including open-access SVG data. To investigate whether their semantic understanding ability is due to potential data leakage, we propose a semantic consistency test by introducing global translations or rotations to SVG graphics, ensuring SE(2) invariance. Such spatial interventions greatly alter the code representation, as SVG graphics consist of lines and Bezier curves with anchor points, and SE(2) operations change all numerical values in the code. However, the SVG's semantics—such as shape or color—remain unaffected by this perturbation. This allows us to examine how LLMs behave when the same vector graphics are presented with drastic code-numerical changes (see Appendix A.1 for details). Note that we perform non-trivial coordinate-level perturbations to the code, rather than using SVG transformation functions, to prevent shortcut learning by LLMs. Due to the nested structure of the tested SVG code, we visually inspect the perturbed renderings to ensure that the semantics remain unchanged after perturbation. If the model maintains the performance consistency under these perturbations, it suggests that its semantic understanding is based on a fundamental level of comprehension or visual imagery rather than a trivial memorization.

Dataset specifics. We use our SVG dataset to evaluate the semantic consistency with respect to translation and rotation. For each SVG sample, we randomly choose 5 different translations (T) and rotations plus translations (SE(2), harder case), resulting in a visually small amount of spatial shifts of the rendered object, meaning nearly no changes in semantics, but **complete** change in SVG code numeric given the shift of SVG's anchor points and curves. Then we evaluate all the LLMs with the same question set of the SVG-Understanding benchmark but with these perturbed code inputs.

Model	SVG - Understanding						SVG - Invariance				CAD			
	Avg	Semantics	Count	Color	Shape	Reason	T Avg.	SE(2) Avg.	T Cons.	SE(2) Cons.	Avg	3D	3D$_{\text{complex}}$	2D
Open-source generic LLM														
Gemma-1.1-2B	0.317	0.321	0.333	0.25	0.356	0.287	0.312	0.270	0.954	0.920	0.278	0.294	0.253	0.281
Gemma-1.1-7B	0.393	0.347	0.275	0.453	0.523	0.299	0.403	0.390	0.917	0.894	0.476	0.497	0.464	0.460
InternLM2-7B	0.382	0.279	0.324	0.570	0.431	0.299	0.381	0.381	0.788	0.772	0.480	0.551	0.446	0.411
InternLM2-20B	0.424	0.255	0.379	0.623	0.483	0.276	0.426	0.407	0.777	0.727	0.525	0.586	0.490	0.474
InternLM2.5-7B	0.421	0.273	0.317	0.598	0.515	0.282	0.419	0.404	0.809	0.778	0.562	0.639	0.506	0.509
Yi-1.5-9B	0.355	0.309	0.404	0.493	0.297	0.301	0.372	0.374	0.947	**0.947**	0.469	0.581	0.416	0.361
Yi-1.5-34B	0.443	0.308	0.364	0.644	0.523	0.234	0.446	0.423	0.845	0.819	0.583	0.649	0.563	0.510
Aya-23-8B	0.290	0.244	0.255	0.343	0.326	0.259	0.290	0.273	0.942	0.896	0.428	0.508	0.384	0.359
Aya-23-35B	0.442	0.307	0.354	0.648	0.511	0.318	0.451	0.434	0.898	0.857	0.488	0.551	0.429	0.457
Command R-35B	0.461	0.311	0.442	0.676	0.495	0.341	0.478	0.443	0.833	0.803	0.536	0.579	0.509	0.504
Command R-104B	0.500	0.339	0.449	0.727	0.565	0.341	0.521	0.477	0.917	0.875	0.583	0.634	0.570	0.524
Qwen-1.5-7B	0.376	0.226	0.317	0.563	0.471	0.234	0.371	0.382	0.792	0.780	0.486	0.560	0.426	0.443
Qwen-1.5-32B	0.494	0.307	0.501	0.713	0.552	0.310	0.512	0.492	**0.972**	0.938	0.575	0.664	0.567	0.456
Qwen-1.5-72B	0.466	0.299	0.319	0.698	0.598	0.265	0.474	0.461	0.883	0.854	0.600	0.658	0.590	0.526
Qwen-1.5-110B	0.499	0.324	0.431	0.734	0.560	0.332	0.486	0.470	0.839	0.821	0.632	0.711	0.607	0.546
Qwen-2-72B	0.537	0.373	0.426	0.770	0.630	0.372	0.520	0.491	0.869	0.852	0.692	0.753	0.669	0.630
Mistral-7B v0.3	0.417	0.304	0.324	0.624	0.470	0.296	0.434	0.417	0.919	0.895	0.495	0.551	0.481	0.429
Mistral-NeMo-12B	0.449	0.296	0.355	0.652	0.548	0.296	0.480	0.443	0.894	0.855	0.568	0.623	0.549	0.510
Mistral-Large2-123B	0.572	0.389	0.558	0.814	0.635	0.408	0.577	0.540	0.889	0.847	0.710	0.755	0.716	0.640
LLama3-8B	0.429	0.304	0.372	0.626	0.484	0.293	0.426	0.410	0.905	0.873	0.550	0.633	0.472	0.512
LLama3-70B	0.548	0.364	0.496	0.749	0.645	0.369	0.559	0.525	0.905	0.874	0.634	0.694	0.619	0.566
LLama3.1-8B	0.465	0.339	0.385	0.667	0.533	0.268	0.464	0.448	0.821	0.806	0.574	0.626	0.539	0.534
LLama3.1-70B	0.574	0.400	0.543	0.788	0.659	0.411	0.584	0.554	0.856	0.825	0.688	0.739	0.663	0.641
LLama3.1-405B	0.580	0.376	**0.584**	0.816	0.647	0.389	0.570	0.548	0.840	0.817	0.717	0.767	0.700	0.661
Open-source code LLM														
CodeQwen1.5-7B	0.301	0.245	0.262	0.344	0.387	0.245	0.327	0.324	0.883	0.859	0.376	0.419	0.350	0.340
DeepSeek-Coder-V2-16B	0.451	0.309	0.379	0.637	0.548	0.268	0.496	0.476	0.902	0.867	0.547	0.611	0.521	0.481
Codestral-22B-v0.1	0.491	0.309	0.446	0.698	0.581	0.321	0.503	0.470	0.860	0.816	0.602	0.659	0.577	0.547
Proprietary models														
GPT-3.5 Turbo	0.498	0.319	0.451	0.729	0.577	0.338	0.509	0.492	0.897	0.870	0.576	0.654	0.530	0.510
GPT-4 Turbo	0.609	0.764	0.539	0.832	0.687	0.412	0.606	0.576	0.867	0.835	0.716	0.762	0.694	0.674
GPT-4o mini	0.585	0.398	0.504	0.791	0.709	0.414	0.595	0.561	0.881	0.852	0.659	0.737	0.612	0.594
GPT-4o	0.633	**0.787**	0.553	0.832	0.696	0.471	0.625	0.586	0.878	0.844	0.733	**0.782**	0.711	0.686
Claude 3 Haiku	0.486	0.264	0.398	0.750	0.610	0.301	0.496	0.476	0.902	0.867	0.612	0.677	0.581	0.549
Claude 3 Sonnet	0.565	0.375	0.503	0.803	0.657	0.395	0.582	0.566	0.875	0.838	0.644	0.673	0.649	0.599
Claude 3.5 Sonnet	**0.674**	0.505	**0.584**	**0.891**	**0.758**	**0.527**	**0.670**	**0.649**	0.903	0.870	**0.742**	0.769	**0.727**	**0.717**

Table 1: Performance of various LLMs on SGP-Bench. This table evaluates how well models understand SVG inputs ('SVG - Understanding') and their behavior under random perturbations of these inputs ('SVG - Invariance'). It also assesses 3D & 2D semantic understanding of CAD code. We found the results demonstrate the "scaling law", with larger LLMs in the same family showing superior performance. Bold texts indicates performance with **1st rank**, and underlined texts indicates performance with 2nd rank and 3rd rank.

Evaluation. We measure the semantic consistency with two metrics: 1) the average accuracy of all perturbed SVG inputs' question-answering accuracy, showing the overall accuracy once the samples are intervened; and 2) the proposed "consistency score" that first counts the most common selection for each group of perturbed samples (where they were translated or rotated from the same SVG program), then divided by the number of perturbations. This score indicates how much the LLMs being consistent (no matter right or wrong) regardless of the drastic program change. If the score is close to 1, it means all the predictions are the same even with totally different input codes, vice versa.

Experimental results and discussion. Our experiments with the SVG-Invariance benchmark demonstrate that most LLMs exhibit robust semantic understanding of graphics programs under translation (T) and translation + rotation (SE(2)) perturbations. In Table 1, most of the LLMs achieve over 80% consistency in both perturbation settings, with half of the models exceeding 90% consistency. Not only do the models remain consistent in their predictions under perturbations, but their performance on perturbed inputs also shows minimal fluctuation compared to their performance on the SVG-Understanding benchmark. We posit that this indicates that the semantic understanding ability that we evaluate of LLMs is unlikely due to data leakage, but rather, could stem from a potential foundational capability to interpret the semantics of deterministic, symbolic graphics programs.

4.4 Prediction Entropy of LLMs and Humans

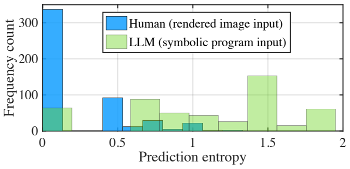
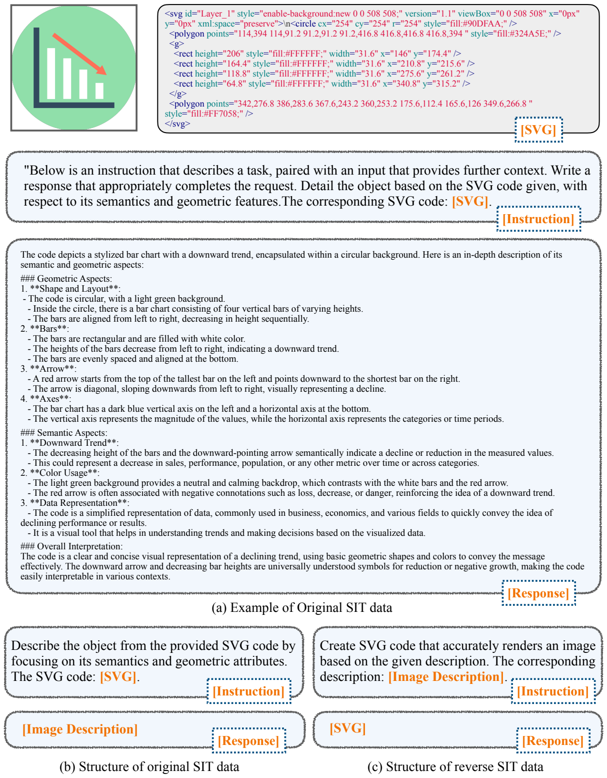


Figure 11: Comparison of prediction entropy.

To study the consensus of LLMs, we compute the average prediction entropy on 500 symbolic programs using GPT-4o, LLama3-8B, LLama3-70B, Mistral-7B, Yi-1.5-34B, Gemma-1.1-7B and Qwen-1.5-72B. We conduct a human experiment on the rendered images of these programs and collect the answers (each question has at least 5 participants, see Appendix B). Figure 11 shows that humans have strong consensus when answering questions based on images, while LLMs show low consensus when answering questions based on symbolic programs. This implies that LLMs may have different inner working mechanisms to understand symbolic programs. We are excited by future work to better investigate this difference.

5 Improving LLMs with Symbolic Instruction Tuning

Generating symbolic instruction data. Inspired by how visual instruction tuning [56] enables large vision-language models to understand images with visual-question-answering (VQA) data, we design a new method to perform symbolic instruction tuning for LLMs to better bridge the gap between the semantic understanding and symbolic reasoning over the graphics programs. To our knowledge, there exist no semantic instruction datasets directly over symbolic graphics programs, After rendering these symbolic graphics programs into images, we can easily query powerful vision-language models (*e.g.*, GPT-4o is used in our case) to obtain a detailed semantic captioning based on the rendered image. The intuition is straightforward, as we want to build an intrinsic connection between semantic description and symbolic programs. The instruction data is created in a similar spirit to our benchmark. We leverage the correspondence between symbolic problems and graphics content, and then use the rendered images to obtain semantically rich description. Following this idea, we construct the first semantic description dataset for symbolic graphics programs. Specifically, for each image rendered from a symbolic graphics program, we prompt GPT-4o to produce a semantically-rich description. Finally, we collect a dataset that contains detailed semantic descriptions for 72K symbolic programs. Moreover, our SIT data can also be used in a reverse fashion (rev-SIT), *i.e.*, rephrasing the answer as the new question and the question as the new answer. Figure 12 shows the comparison between original and reverse SIT data.



```
<svg id="Layer_1" style="enable-background:new 0 0 508 508;" version="1.1" viewBox="0 0 508 508" x="0px"
y="0px" xml:space="preserve">\n<circle cx="254" cy="254" r="254" style="fill:#90DFAA;" />
  <polygon points="114,394 114,91.2 91.2,91.2 91.2,416.8 416.8,416.8 416.8,394 " style="fill:#324A5E;" />
  <g>
    <rect height="206" style="fill:#FFFFFF;" width="31.6" x="146" y="174.4" />
    <rect height="164.4" style="fill:#FFFFFF;" width="31.6" x="210.8" y="215.6" />
    <rect height="118.8" style="fill:#FFFFFF;" width="31.6" x="275.6" y="261.2" />
    <rect height="64.8" style="fill:#FFFFFF;" width="31.6" x="340.8" y="315.2" />
  </g>
  <polygon points="342,276.8 386,283.6 367.6,243.2 360,253.2 175.6,112.4 165.6,126 349.6,266.8 "
style="fill:#FF7058;" />
</svg>
```

"Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Detail the object based on the SVG code given, with respect to its semantics and geometric features.The corresponding SVG code: [SVG].

```
The code depicts a stylized bar chart with a downward trend, encapsulated within a circular background. Here is an in-depth description of its
semantic and geometric aspects:
### Geometric Aspects:
1. **Shape and Layout**:
 - The code is circular, with a light green background.
   - Inside the circle, there is a bar chart consisting of four vertical bars of varying heights.
   - The bars are aligned from left to right, decreasing in height sequentially.
2. **Bars**:
   - The bars are rectangular and are filled with white color.
   - The heights of the bars decrease from left to right, indicating a downward trend.
   - The bars are evenly spaced and aligned at the bottom.
3. **Arrow**:
   - A red arrow starts from the top of the tallest bar on the left and points downward to the shortest bar on the right.
   - The arrow is diagonal, sloping downwards from left to right, visually representing a decline.
4. **Axes**:
   - The bar chart has a dark blue vertical axis on the left and a horizontal axis at the bottom.
   - The vertical axis represents the magnitude of the values, while the horizontal axis represents the categories or time periods.
### Semantic Aspects:
1. **Downward Trend**:
   - The decreasing height of the bars and the downward-pointing arrow semantically indicate a decline or reduction in the measured values.
   - This could represent a decrease in sales, performance, population, or any other metric over time or across categories.
2. **Color Usage**:
   - The light green background provides a neutral and calming backdrop, which contrasts with the white bars and the red arrow.
   - The red arrow is often associated with negative connotations such as loss, decrease, or danger, reinforcing the idea of a downward trend.
3. **Data Representation**:
   - The code is a simplified representation of data, commonly used in business, economics, and various fields to quickly convey the idea of
declining performance or results.
   - It is a visual tool that helps in understanding trends and making decisions based on the visualized data.
### Overall Interpretation:
The code is a clear and concise visual representation of a declining trend, using basic geometric shapes and colors to convey the message
effectively. The downward arrow and decreasing bar heights are universally understood symbols for reduction or negative growth, making the code
easily interpretable in various contexts.
```

(a) Example of Original SIT data

Describe the object from the provided SVG code by focusing on its semantics and geometric attributes. The SVG code: [SVG].

[Image Description]

(b) Structure of original SIT data

Create SVG code that accurately renders an image based on the given description. The corresponding description: [Image Description].

[SVG]

(c) Structure of reverse SIT data

Figure 12: Comparison between original and reverse SIT data.

Supervised finetuning with symbolic instruction data. We generally follow the standard instruction fine-tuning procedure (including the default hyperparameter settings) from Alpaca [93] and use supervised finetuning to train open-source LLMs with our own symbolic instruction data. To facilitate future research, our symbolic instruction data is also made publicly available.

Experimental results. We perform supervised finetuning on Llama-3.1-8B with orthogonal finetuning [60, 75] to demonstrate the effectiveness of SIT. Here we use the original SIT data (no rev-SIT data is used). In Appendix E.2, we provide results on both Llama-3-8B and Gemma-7B to show that the performance gain is agnostic to the base LLM. The performance of LoRA [36] is also given in Appendix E.2 (only slightly worse than OFT). From the experimental results in Table 2, we observe that SIT has consistently improved the semantic graphics program understanding of LLMs, increasing the performance of Llama-3.1-8B from 46.7% to 51.4% with 55K instruction question-answer pairs. With more instruction data (from 10K to 55K), the performance also increases. We note that Llama-3.1-8B achieves competitive performance among all open-source LLMs after being finetuned with SIT. The performance is already better than GPT-3.5t. The finetuning results demonstrates that the ability of symbolic graphics program understanding can be improved with SIT. However, the improved performance of Llama-3.1-8B remains worse than Llama-3.1-70B, indicating that the tested ability is fundamental and differences between models of varying scales cannot be easily leveled by finetuning on benchmark-like data.

Dataset Size	Accuracy
Original	46.5
SIT-10k	48.0 (+1.3)
SIT-25k	50.3 (+3.6)
SIT-40k	51.2 (+4.5)
SIT-55k	**51.4** (+4.7)

Table 2: Performance of SIT.

6 SIT Can Improve General Reasoning Ability

Since we show in Figure 4 that the ability to understand symbolic graphics programs is associated with some fundamental reasoning abilities of LLMs, we are particularly interested in whether symbolic graphcis programs can be used as a novel data source for building better instruction tuning datasets, which can help to improve the general reasoning ability of LLMs. To verify this, we test the instruction-tuned models on a variety of popular LLM benchmarks, including benchmarks focusing on natural language understanding (XNLI [17], IFEval [119], HellaSwag [117], C-Eval [39], CoQA [80], MMLU [34], SQuAD2.0 [77]), generic reasoning (BigBenchHard [92], PIQA [7], AGIEval [118]) and mathematical reasoning (Arithmetic [9], MathQA [3], GSM8k [14], ASDiv [71]).

Experimental results and discussion. We use the Llama-3.1-8B model (without instruction tuning), and the baseline is finetuned with Open-Instruct (OI) [97] that contains 143K question-answer pairs (details in Appendix E.1). We evaluate whether finetuning with SIT data can improve general reasoning by testing three ways of using SIT data: (1) mixing original SIT data into OI; (2) mixing the reverse SIT data into OI; (3) mixing both original and reverse SIT data into OI. The results are given in Table 3. We can observe that mixing SIT data can generally improve the instruction following and the reverse usage of SIT data (*i.e.*, symbolic graphics program generation) can improve a set of reasoning abilities that are complementary to symbolic graphics program understanding. The mixture of both original and reverse SIT data often achieves better performance than the OI baseline, the OI + SIT baseline and the OI + rev-SIT baseline. These results are consistent with recent findings that training on code can enhance reasoning ability [66, 4] and mathematical understanding [88]. Symbolic graphics programs, a specialized form of code, are used to generate visual graphics content. Like traditional code, they possess a hierarchical structure, but unlike typical programs that produce numerical outputs, symbolic graphics programs generate outputs rich in semantic information, encompassing multiple challenging reasoning tasks such as component localization, color identification, affordance prediction, and semantic and geometric understanding. For instance, answering a question like “What is the object primarily used for?” requires LLMs to first semantically identify the object and then determine its usage. This process involves multiple interconnected reasoning steps, where an error in any one of them leads to an incorrect final answer. SIT enhances reasoning abilities by interpreting low-level graphics programs through high-level natural language descriptions. From Figure 12(a), we see that symbolic graphics program descriptions are highly detailed and semantic—qualities often lacking in general programs.

Benchmark	OI	OI-SIT	OI-rev-SIT	OI-mixed-SIT
XNLI	41.8	**43.3** (+1.5)	43.1 (+1.3)	42.9 (+1.1)
IFEval$_{\text{prompt}}$	14.8	16.3 (+1.5)	**18.3** (+3.5)	16.6 (+1.8)
IFEval$_{\text{inst.}}$	24.9	28.9 (+4.0)	**30.5** (+5.6)	29.6 (+4.7)
HellaSwag	60.0	60.2 (+0.2)	**60.5** (+0.5)	60.4 (+0.4)
C-Eval	46.4	47.9 (+1.5)	48.0 (+1.6)	**48.1** (+1.7)
MMLU	60.4	61.0 (+0.6)	61.1 (+0.7)	**61.6** (+1.2)
SQuAD2.0	28.9	28.7 (-0.2)	**31.6** (+2.7)	29.9 (+1.0)
BBH	59.5	60.7 (+1.2)	60.2 (+0.7)	**61.2** (+1.7)
PIQA	79.9	80.3 (+0.4)	80.3 (+0.4)	**80.4** (+0.5)
AGIEval	23.7	30.3 (+6.6)	**31.6** (+7.9)	29.2 (+5.5)
Arithmetic	89.8	**91.8** (+2.0)	90.1 (+0.3)	**91.8** (+2.0)
MathQA	39.3	40.4 (+1.1)	40.3 (+1.0)	**40.7** (+1.4)
GSM8k	48.2	50.7 (+2.5)	51.0 (+2.8)	**51.5** (+3.3)
CoQA	67.9	**69.1** (+1.2)	68.7 (+0.8)	**69.1** (+1.2)
ASDiv	18.5	**21.8** (+3.3)	20.1 (+1.6)	21.3 (+2.8)

Table 3: Results on a variety of popular LLM evaluation benchmarks when performing instruction tuning with or without SIT. The Open-Instruct (OI) dataset serves as our baseline.

7 A critical View on current LLM’s Capability

Despite the observed remarkable capability of LLMs to perform complex, multi-step reasoning over symbolic programs, it is evident that there remains substantial potential for further advancements. We provide an intriguing experiment to demonstrate that SVG programs can be quite difficult for LLMs to understand such that even if the corresponding rendered images are fairly easy for humans to recognize, all these powerful LLMs still fail dramatically.

Method	Accuracy
LLama3-70B	10.0
Qwen-1.5-110b	10.0
Qwen-2-70b	11.3
GPT-3.5t	10.2
GPT-4t	10.6
GPT-4o	**13.0**

Table 4: Accuracy of LLMs on SGP-MNIST.

Specifically, we construct symbolic graphics programs that can produce MNIST-like images, as shown in Figure 4 (and Appendix A.1). Our SGP-MNIST dataset contains 1,000 symbolic graphics programs (100 per digit), each asking which digit the SVG program represents. The results are given in Table 4. Even the powerful GPT-4o can only achieve an accuracy slightly higher than the chance-level. The MNIST-like symbolic program presents a unique challenge due to the absence of semantic components for LLMs to reason upon. Instead, it comprises convoluted, irregular path trajectories that resemble handwritten digits. Additionally, the program contains not only single paths but enclosed loops to represent the “thickness” of digits, demanding precise path planning by the LLMs. For instance, the digit 1 is not represented as a single “line” but rather as an elongated loop, which must be distinguished from more oval-shaped loops, such as those representing the digit 0.

Without prior knowledge of digit "thickness," the LLM must infer this distinction through detailed reasoning over the loop structures, further elevating the complexity of the task. The chance-level performance suggests that how LLMs understand SVG programs is very different from how humans understand images; better understanding similarities and differences in human and machine reasoning is important if we are to build systems that can appropriately work with us [16]. There are many exciting yet totally unexplored problems in this task, and our benchmark can serve as a stepping stone to improving symbolic graphics program understanding for LLMs.

8 Related Work and Concluding Remarks

Symbolic graphics programs. Generating visual data by procedural modeling with symbolic programs has been essential to computer graphics since its inception, particularly for 2D shapes and 3D geometry. See [83] for an overview. Common program types include constructive-solid geometry (CSG) [21, 45, 81, 89, 112], computer-aided design (CAD) [30, 50, 51, 87, 108], vector graphics (*e.g.*, SVG) [78, 79], L-systems [33], and customized domains [22, 95, 18, 37, 24, 69]. Among these, SVGs are constructed from primitive shapes like vector paths, curves, or polygons. Central to SVGs is the vector path, providing detailed control over graphics and geometry primitives. Similarly, procedural 3D geometric modeling, particularly in CAD applications, involves parameterized operations to produce geometry. Datasets like the ABC [47] and Fusion 360 Gallery [102] offer hierarchical decomposition, joints, contact surfaces, construction sequences, and shape segmentation based on modeling operations. Our paper focuses on graphics programs of SVG and CAD by introducing a new semantic understanding task that requires a challenging reasoning over the programs.

Graphics program understanding and generation. As graphics programs often provide compact, scalable and potentially more semantic descriptions compared to raw pixels and voxels, it has been widely explored to discover graphics programs for 2D images like 2D hand drawings and synthetic patterns [98, 99, 89, 23, 82, 29, 87], for 3D objects represented in voxels and meshes [29, 43, 95, 8, 102, 89, 23] and for 3D scenes represented by multi-view images [68, 49, 104, 111, 69, 61, 53, 31, 30, 46, 20]. [104] infers custom-designed markup code from images that can be easily translated to renderer-friendly inputs. In follow-up work, [111] explore how graphics programs can be used for visual question answering (VQA). Recently, [48] has advanced this direction by examining large language models (LLMs) for synthesizing graphics programs to reconstruct visual input. In contrast, we benchmark LLMs to perform semantic-level question answering, similar to VQA, but use graphics programs as input without relying on any visual modality.

Large language models. LLMs have demonstrated growing potential in many applications, ranging from mathematical problem solving and theorem proving assistance [65, 114, 116, 15] to aiding biological discovery [67, 27, 94, 59]. Applying LLMs for programming tasks is also a popular direction of research. Specifically, many works have explored topics such as code retrieval [26], automated testing [19, 62, 109], repairing [44, 106, 41, 42, 100], documentation [13, 2], and generation [11, 5, 54, 28, 72]. These abilities of understanding and generating programs are usually gained from pretraining or finetuning on large datasets of code. Our work investigates LLMs' capability of understanding symbolic graphics programs, which differs significantly from the prior works since the semantic meaning of graphics programs are often defined visually by their corresponding graphics.

Relevant benchmarks and datasets. Many benchmarks have evaluated different aspects of LLMs: AI safety/ethics [107, 38], out-of-distribution performance [115, 110], API/tool usage [52], code generation [35], *etc*. Perhaps the most relevant aspect of LLM evaluation to our task is (non-graphics) program understanding abilities [96, 44, 35, 64, 57, 63, 90, 34, 11]. As graphics programs can be rendered into images, it is also highly relevant to investigate how vision-language models are capable of visual understanding [12, 1, 74, 40, 70, 32, 6, 91, 85, 120]. For SVG programs, [10] studies whether LLMs can understand them and [122] introduces a concurrent benchmark for this purpose. Different from existing benchmarks, SGP-Bench is one of the first benchmarks to evaluate the semantic understanding of general graphics programs.

Concluding Remarks. We propose a new task over which to evaluate LLMs to contribute towards a multi-faceted assessment of their capabilities: the task of "understand" images – without image; that is, directly from symbolic graphics programs. For such a task, we create SGP-Bench, a benchmark that shows distinguishable results between LLMs and introduce the method of Symbolic Instruction Finetuning (SIT) that improves LLMs' capability of understanding graphics programs. More importantly, SIT also helps to improve general reasoning ability of LLMs.

REFERENCES

[1] Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. Nocaps: Novel object captioning at scale. In *ICCV*, 2019. 11

[2] Toufique Ahmed and Premkumar T. Devanbu. Few-shot training llms for project-specific code-summarization. In *ASE*, 2022. 11

[3] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. Mathqa: Towards interpretable math word problem solving with operation-based formalisms, 2019. 10

[4] Viraat Aryabumi, Yixuan Su, Raymond Ma, Adrien Morisot, Ivan Zhang, Acyr Locatelli, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. To code, or not to code? exploring impact of code in pre-training. *arXiv preprint arXiv:2408.10914*, 2024. 10

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. 1, 11

[6] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *UIST*, 2010. 11

[7] Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020. 10

[8] Martin Bokeloh, Michael Wand, and Hans-Peter Seidel. A connection between partial symmetry and inverse procedural modeling. In *SIGGRAPH*, 2010. 11

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf. 10

[10] Mu Cai, Zeyi Huang, Yuheng Li, Haohan Wang, and Yong Jae Lee. Leveraging large language models for scalable vector graphics-driven image understanding. *arXiv preprint arXiv:2306.06094*, 2023. 11

[11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 11

[12] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015. 11

[13] Colin B Clement, Dawn Drain, Jonathan Timcheck, Alexey Svyatkovskiy, and Neel Sundaresan. Pymt5: multi-mode translation of natural language and python code with transformers. *arXiv preprint arXiv:2010.03150*, 2020. 11

[14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. 10

[15] Katherine M Collins, Albert Q Jiang, Simon Frieder, Lionel Wong, Miri Zilka, Umang Bhatt, Thomas Lukasiewicz, Yuhuai Wu, Joshua B Tenenbaum, William Hart, et al. Evaluating language models for mathematics through interactions. *PNAS*, 2024. 11

[16] Katherine M. Collins, Ilia Sucholutsky, Umang Bhatt, Kartik Chandra, Lionel Wong, Mina Lee, Cedegao E. Zhang, Tan Zhi-Xuan, Mark Ho, Vikash Mansinghka, Adrian Weller, Joshua B. Tenenbaum, and Thomas L. Griffiths. Building machines that learn and think with people, 2024. URL https://arxiv.org/abs/2408.03943. 11

[17] Alexis Conneau, Guillaume Lample, Ruty Rinott, Adina Williams, Samuel R Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. *arXiv preprint arXiv:1809.05053*, 2018. 10

[18] Boyang Deng, Sumith Kulal, Zhengyang Dong, Congyue Deng, Yonglong Tian, and Jiajun Wu. Unsupervised learning of shape programs with repeatable implicit parts. In *NeurIPS*, 2022. 11

[19] Yinlin Deng, Chunqiu Steven Xia, Chenyuan Yang, Shizhuo Dylan Zhang, Shujing Yang, and Lingming Zhang. Large language models are edge-case fuzzers: Testing deep learning libraries via fuzzgpt. *arXiv preprint arXiv:2304.02014*, 2023. 11

[20] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Meta-Sim2: Unsupervised learning of scene structure for synthetic data generation. In *ECCV*, 2020. 11

[21] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. InverseCSG: Automatic conversion of 3D models to CSG trees. *ACM Transactions on Graphics*, 37(6), dec 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275006. URL https://doi.org/10.1145/3272127.3275006. 11

[22] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Josh Tenenbaum. Learning to infer graphics programs from hand-drawn images. In *NeurIPS*, 2018. 11

[23] Kevin Ellis, Maxwell Nye, Yewen Pu, Felix Sosa, Josh Tenenbaum, and Armando Solar-Lezama. Write, execute, assess: Program synthesis with a REPL. In *NeurIPS*, 2019. 11

[24] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *PLDI*, 2021. 11

[25] Kevin Ellis, Lionel Wong, Maxwell Nye, Mathias Sable-Meyer, Luc Cary, Lore Anaya Pozo, Luke Hewitt, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: growing generalizable, interpretable knowledge with wake–sleep bayesian program learning. *Philosophical Transactions of the Royal Society A*, 381(2251), 2023. 2

[26] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020. 11

[27] Noelia Ferruz and Birte Höcker. Controllable protein design with language models. *Nature Machine Intelligence*, 4(6):521–532, 2022. 11

[28] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Scott Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis. In *ICLR*, 2023. 11

[29] Aditya Ganeshan, R. Kenny Jones, and Daniel Ritchie. Improving unsupervised visual program inference with code rewriting families. In *ICCV*, 2023. 11

[30] Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S.M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. In *ICML*, 2018. 11

[31] Nishad Gothoskar, Marco Cusumano-Towner, Ben Zinberg, Matin Ghavamizadeh, Falk Pollok, Austin Garrett, Josh Tenenbaum, Dan Gutfreund, and Vikash Mansinghka. 3DP3: 3D scene perception via probabilistic programming. In *NeurIPS*, 2021. 11

[32] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017. 11

[33] Jianwei Guo, Haiyong Jiang, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, Dani Lischinski, and Hui Huang. Inverse procedural modeling of branching structures by inferring L-systems. *ACM Transactions on Graphics*, 39(5), jun 2020. ISSN 0730-0301. URL https://doi.org/10.1145/3394105. 11

[34] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. 10, 11, 22

[35] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *NeurIPS*, 2021. 11

[36] Edward J. Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9. 9, 39

[37] Yiwei Hu, Chengan He, Valentin Deschaintre, Julie Dorsey, and Holly Rushmeier. An inverse procedural modeling pipeline for SVBRDF maps. *ACM Transactions on Graphics*, 41(2), 2022. 11

[38] Yue Huang, Qihui Zhang, Lichao Sun, et al. Trustgpt: A benchmark for trustworthy and responsible large language models. *arXiv preprint arXiv:2306.11507*, 2023. 11

[39] Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, Jiayi Lei, Yao Fu, Maosong Sun, and Junxian He. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. *arXiv preprint arXiv:2305.08322*, 2023. 10

[40] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, 2019. 11

[41] Nan Jiang, Kevin Liu, Thibaud Lutellier, and Lin Tan. Impact of code language models on automated program repair. In *ICSE*, 2023. 11

[42] Matthew Jin, Syed Shahriar, Michele Tufano, Xin Shi, Shuai Lu, Neel Sundaresan, and Alexey Svyatkovskiy. Inferfix: End-to-end program repair with llms. In *ESEC/FSE*, 2023. 11

[43] R. Kenny Jones, Homer Walke, and Daniel Ritchie. PLAD: Learning to infer shape programs with pseudo-labels and approximate distributions. In *CVPR*, 2022. 11

[44] Aditya Kanade, Petros Maniatis, Gogul Balakrishnan, and Kensen Shi. Learning and evaluating contextual embedding of source code. In *ICML*, 2020. 11

[45] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. UCSG-NET–Unsupervised discovering of constructive solid geometry tree. In *NeurIPS*, 2020. 11

[46] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-Sim: Learning to generate synthetic datasets. In *ICCV*, 2019. 11

[47] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A big CAD model dataset for geometric deep learning. In *CVPR*, 2019. 11, 21

[48] Peter Kulits, Haiwen Feng, Weiyang Liu, Victoria Abrevaya, and Michael J Black. Re-thinking inverse graphics with large language models. *arXiv preprint arXiv:2404.15228*, 2024. 11

[49] Tejas D. Kulkarni, Pushmeet Kohli, Joshua B. Tenenbaum, and Vikash Mansinghka. Picture: A probabilistic programming language for scene perception. In *CVPR*, 2015. 11

[50] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Sketch2CAD: Sequential CAD modeling by sketching in context. *ACM Transactions on Graphics*, 39(6), nov 2020. ISSN 0730-0301. doi: 10.1145/3414685.3417807. URL https://doi.org/10.1145/3414685.3417807. 11

[51] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2CAD: Parsing freehand drawings into CAD commands. *ACM Transactions on Graphics*, 2022. 11

[52] Minghao Li, Feifan Song, Bowen Yu, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*, 2023. 11

[53] Yikai Li, Jiayuan Mao, Xiuming Zhang, Bill Freeman, Josh Tenenbaum, Noah Snavely, and Jiajun Wu. Multi-plane program induction with 3D box priors. In *NeurIPS*, 2020. 11

[54] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022. 11

[55] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 3

[56] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 3, 9

[57] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023. 11

[58] Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2024. 1

[59] Shengchao Liu, Jiongxiao Wang, Yijin Yang, Chengpeng Wang, Ling Liu, Hongyu Guo, and Chaowei Xiao. Conversational drug editing using retrieval and domain feedback. In *ICLR*, 2024. 11

[60] Weiyang Liu, Zeju Qiu, Yao Feng, Yuliang Xiu, Yuxuan Xue, Longhui Yu, Haiwen Feng, Zhen Liu, Juyeon Heo, Songyou Peng, Yandong Wen, Michael J. Black, Adrian Weller, and Bernhard Schölkopf. Parameter-efficient orthogonal finetuning via butterfly factorization. In *ICLR*, 2024. 9, 39

[61] Yunchao Liu, Jiajun Wu, Zheng Wu, Daniel Ritchie, William T. Freeman, and Joshua B. Tenenbaum. Learning to describe scenes with programs. In *ICLR*, 2019. 11

[62] Zhe Liu, Chunyang Chen, Junjie Wang, Xing Che, Yuekai Huang, Jun Hu, and Qing Wang. Fill in the blank: Context-aware automated text input generation for mobile GUI testing. In *ICSE*, 2023. 11

[63] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*, 2023. 11

[64] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin B. Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie Liu. Codexglue: A machine learning benchmark dataset for code understanding and generation. In *NeurIPS*, 2021. 11

[65] Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023. 11

[66] Yingwei Ma, Yue Liu, Yue Yu, Yuanliang Zhang, Yu Jiang, Changjian Wang, and Shanshan Li. At which training stage does code data help llms reasoning? *arXiv preprint arXiv:2309.16298*, 2023. 10

[67] Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023. 11

[68] Vikash K. Mansinghka, Tejas D. Kulkarni, Yura N. Perov, and Josh Tenenbaum. Approximate Bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, 2013. 11

[69] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *ICLR*, 2019. 11

[70] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *CVPR*, 2019. 11

[71] Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers, 2021. 10

[72] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. In *ICLR*, 2023. 11

[73] Stefan Palan and Christian Schitter. Prolific.ac—a subject pool for online experiments. *Journal of Behavioral and Experimental Finance*, 17:22–27, 2018. 26

[74] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, 2015. 11

[75] Zeju Qiu, Weiyang Liu, Haiwen Feng, Yuxuan Xue, Yao Feng, Zhen Liu, Dan Zhang, Adrian Weller, and Bernhard Schölkopf. Controlling text-to-image diffusion by orthogonal finetuning. *Advances in Neural Information Processing Systems*, 36:79320–79362, 2023. 9, 39

[76] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 3

[77] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018. 10

[78] Pradyumna Reddy, Michael Gharbi, Michal Lukac, and Niloy J. Mitra. Im2Vec: Synthesizing vector graphics without vector supervision. In *CVPR*, 2021. 11

[79] Pradyumna Reddy, Zhifei Zhang, Zhaowen Wang, Matthew Fisher, Hailin Jin, and Niloy Mitra. A multi-implicit neural representation for fonts. In *NeurIPS*, 2021. 11

[80] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. 10

[81] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. ExtrudeNet: Unsupervised inverse sketch-and-extrude for shape parsing. In *ECCV*, 2022. 11

[82] Marzia Riso, Davide Sforza, and Fabio Pellacini. pOp: Parameter optimization of differentiable vector patterns. *Computer Graphics Forum*, 2022. 11

[83] Daniel Ritchie, Paul Guerrero, R. Kenny Jones, Niloy J. Mitra, Adriana Schulz, Karl D. D. Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. *Computer Graphics Forum*, 2023. 11

[84] Joshua S Rule, Joshua B Tenenbaum, and Steven T Piantadosi. The child as hacker. *Trends in cognitive sciences*, 24(11):900–915, 2020. 2

[85] Tanik Saikh, Tirthankar Ghosal, Amish Mittal, Asif Ekbal, and Pushpak Bhattacharyya. Scienceqa: A novel resource for question answering on scholarly articles. *International Journal on Digital Libraries*, 23(3):289–301, 2022. 11

[86] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICML 2020 Workshop on Object-Oriented Learning*, 2020. 6, 21

[87] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P. Adams. Vitruvion: A generative model of parametric CAD sketches. In *ICLR*, 2022. 11

[88] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 10

[89] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural shape parser for constructive solid geometry. In *CVPR*, 2018. 11

[90] Freda Shi, Mirac Suzgun, Markus Freitag, Xuezhi Wang, Suraj Srivats, Soroush Vosoughi, Hyung Won Chung, Yi Tay, Sebastian Ruder, Denny Zhou, et al. Language models are multilingual chain-of-thought reasoners. *arXiv preprint arXiv:2210.03057*, 2022. 11

[91] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *CVPR*, 2019. 11

[92] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, , and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*, 2022. 10

[93] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023. 9

[94] Arun James Thirunavukarasu, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. Large language models in medicine. *Nature Medicine*, 2023. 11

[95] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to infer and execute 3D shape programs. In *ICLR*, 2019. 11

[96] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *CHI*, 2022. 11

[97] VMware AI Labs. Open-instruct. Huggingface.co, 2023. URL https://huggingface.co/datasets/VMware/open-instruct. Accessed: 2024-10-02. 10

[98] O. Št’ava, B. Beneš, R. Měch, D. G. Aliaga, and P. Krištof. Inverse procedural modeling by automatic generation of L-systems. *Computer Graphics Forum*, 2010. 11

[99] O. Št’ava, S. Pirk, J. Kratt, B. Chen, R. Měch, O. Deussen, and B. Benes. Inverse procedural modelling of trees. *Computer Graphics Forum*, 2014. 11

[100] Yuxiang Wei, Chunqiu Steven Xia, and Lingming Zhang. Copiloting the copilots: Fusing large language models with completion engines for automated program repair. In Satish Chandra, Kelly Blincoe, and Paolo Tonella (eds.), *ESEC/FSE*, 2023. 11

[101] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4), 2021. 6, 21

[102] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Transactions on Graphics*, 2021. 11

[103] Lionel Wong, Gabriel Grand, Alexander K Lew, Noah D Goodman, Vikash K Mansinghka, Jacob Andreas, and Joshua B Tenenbaum. From word models to world models: Translating from natural language to the probabilistic language of thought. *arXiv preprint arXiv:2306.12672*, 2023. 2

[104] Jiajun Wu, Joshua B. Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *CVPR*, 2017. 11

[105] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6772–6782, 2021. 6, 21

[106] Chunqiu Steven Xia and Lingming Zhang. Less training, more repairing please: revisiting automated program repair via zero-shot learning. In *ESEC/FSE*, 2022. 11

[107] Guohai Xu, Jiayi Liu, Ming Yan, Haotian Xu, Jinghui Si, Zhuoran Zhou, Peng Yi, Xing Gao, Jitao Sang, Rong Zhang, et al. Cvalues: Measuring the values of chinese large language models from safety to responsibility. *arXiv preprint arXiv:2307.09705*, 2023. 11

[108] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D.D. Willis, and Daniel Ritchie. Inferring CAD modeling sequences using zone graphs. In *CVPR*, 2021. 11

[109] Chenyuan Yang, Yinlin Deng, Runyu Lu, Jiayi Yao, Jiawei Liu, Reyhaneh Jabbarvand, and Lingming Zhang. White-box compiler fuzzing empowered by large language models. *arXiv preprint arXiv:2310.15991*, 2023. 11

[110] Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. Glue-x: Evaluating natural language understanding models from an out-of-distribution generalization perspective. *arXiv preprint arXiv:2211.08073*, 2022. 11

[111] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Josh Tenenbaum. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018. 11

[112] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. CAPRI-Net: Learning compact CAD shapes with adaptive primitive assembly. In *CVPR*, 2022. 11

[113] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. 7

[114] Longhui Yu, Weisen Jiang, Han Shi, YU Jincheng, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024. 11

[115] Lifan Yuan, Yangyi Chen, Ganqu Cui, Hongcheng Gao, Fangyuan Zou, Xingyi Cheng, Heng Ji, Zhiyuan Liu, and Maosong Sun. Revisiting out-of-distribution robustness in nlp: Benchmarks, analysis, and llms evaluations. In *NeurIPS*, 2024. 11

[116] Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023. 11

[117] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. 10

[118] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023. 10

[119] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. 10

[120] Luowei Zhou, Chenliang Xu, and Jason Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018. 11

[121] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. In *ICLR*, 2024. 3

[122] Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. Vgbench: Evaluating large language models on vector graphics understanding and generation. *arXiv preprint arXiv:2407.10972*, 2024. 11

Appendix

Table of Contents

A Benchmark Details

We adopt implementations from other projects to build our SGP-Bench. We follow the implemntation of *MathVista*[1] for querying GPT or open-sourced Llama3.1-8B and perform LLM-based answer extraction, *vLLM*[2] for efficient model inference and *simple-evals*[3] for a unified benchmarking framework.

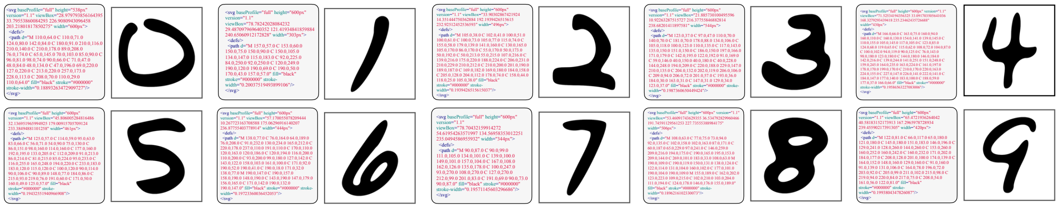
Our data license follows the license of the original license of our data source.

A.1 Data preparation

SGP-Bench (SVG). Our SVG data are sampled from kaggle *SVG Icons* dataset[4] and we build our SGP-Bench (SVG) using text prompts from F.1. The original data from kaggle *SVG Icons* is crawled from *SVGrepo*[5]. The website is merely an aggregator, so it follows that any content on it must at least be licensed permissively enough for SVGrepo to distribute it, and therefore it is acceptable to distribute as part of a collection in our benchmark. Refer to the SVGrepo license for further details.

SVG Invariance: We use beautifulsoup and SvgLib to process the SVG XML code to perform translation and rotation perturbations for the invariance investigation, a visual sample can be found in Fig. 17. Specifically, as we assume that all XML elements in each SVG figure do not possess any "transform" attribute as it complicates the augmentation process. For elements that can be fully specified by coordinates (*e.g.*, <rect>, <polygon>), we perform augmentation by perturbing these coordinates. For <path> elements in which path information is fully specified in "d" attributes, we first turn all relative operations (*e.g.*, "l 2 3", meaning that draw a line from the current position (x, y) to the position $(x + 2, y + 3)$) into absolute ones, and later perturb the coordinates but not other path attributes. As mentioned in the main paper, small spatial perturbations can drastically change the numerics of the SVG XML code (See Section C for more details).

SGP-Bench (CAD). Our CAD (3D) sequences data are sampled from *DeepCAD* [105] datasets, which contains around 180k manually constructed CAD sequences that are originally from the ABC dataset [47]. We manually sample 1000 sequences and use pythonocc (OpenCASCADE) to verify and normalize these CAD sequences, then render the front and back view of the 3D CAD models. Since all the CAD models are from the OnShape platform, the copyright of the CAD models is owned by their creators. For licensing details, see Onshape Terms of Use 1.g.ii. Our CAD ($\text{3D}_{\text{complex}}$) sequences are sampled from *Fusion360 Reconstruction Dataset* [101] datasets, which contains 8,625 sequences, with more complex curve operations for constructing sketches. Our CAD (2D) sequences are sampled from *SketchGraphs* [86] dataset, which consists of 15 million sketches extracted from real-world CAD models. The major difference is it consists of 2D CAD sketches without Extrusion operations.



```
<svg baseProfile="full" height="538px"
version="1.1" viewBox="28.979793856164395
33.79553860084293 226.9080943096458
203.21801817850275" width="600px">
  <defs/>
  <path d="M 110.0,64.0 C 110.0,71.0
124.0,80.0 142.0,84.0 C 180.0,91.0 210.0,116.0
210.0,140.0 C 210.0,178.0 89.0,208.0
76.0,174.0 C 65.0,145.0 70.0,103.0 85.0,90.0 C
96.0,81.0 98.0,74.0 90.0,66.0 C 71.0,47.0
48.0,84.0 48.0,134.0 C 47.0,196.0 69.0,220.0
127.0,220.0 C 213.0,220.0 257.0,173.0
228.0,113.0 C 208.0,70.0 110.0,29.0
110.0,64.0" fill="black" stroke="#000000"
stroke-width="0.18893263472909727"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1"
viewBox="78.78242028084232
29.487097969640352 121.41934841859884
240.65060912172828" width="303px">
  <defs/>
  <path d="M 157.0,57.0 C 153.0,60.0
150.0,75.0 150.0,90.0 C 150.0,105.0
134.0,147.0 115.0,183.0 C 92.0,225.0
84.0,250.0 92.0,250.0 C 120.0,249.0
190.0,120.0 190.0,69.0 C 190.0,50.0
170.0,43.0 157.0,57.0" fill="black"
stroke="#000000" stroke-
width="0.20037519493899106"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="33.90302867421924
14.351444750362884 192.1939426515615
232.92512452536593" width="496px">
  <defs/>
  <path d="M 105.0,38.0 C 102.0,41.0 100.0,51.0
100.0,61.0 C 100.0,73.0 105.0,77.0 115.0,74.0 C
155.0,58.0 179.0,139.0 141.0,160.0 C 130.0,165.0
105.0,170.0 86.0,170.0 C 55.0,170.0 50.0,173.0
50.0,192.0 C 50.0,213.0 55.0,215.0 107.0,216.0 C
139.0,216.0 175.0,220.0 188.0,224.0 C 206.0,231.0
210.0,229.0 210.0,212.0 C 210.0,200.0 201.0,190.0
189.0,187.0 C 169.0,182.0 169.0,180.0 184.0,158.0
C 205.0,128.0 204.0,112.0 178.0,74.0 C 158.0,44.0
118.0,25.0 105.0,38.0" fill="black"
stroke="#000000" stroke-
width="0.1939426515615037"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="21.885730100495596
10.92263287515727 216.37755846882814
238.68201411897581" width="544px">
  <defs/>
  <path d="M 123.0,37.0 C 97.0,47.0 110.0,70.0
140.0,70.0 C 181.0,70.0 178.0,88.0 134.0,106.0 C
105.0,118.0 100.0,123.0 110.0,135.0 C 117.0,143.0
135.0,150.0 151.0,150.0 C 186.0,150.0 197.0,166.0
171.0,179.0 C 142.0,195.0 122.0,192.0 91.0,169.0
C 59.0,146.0 40.0,150.0 40.0,180.0 C 40.0,228.0
144.0,248.0 194.0,209.0 C 220.0,188.0 229.0,147.0
210.0,135.0 C 204.0,132.0 202.0,119.0 206.0,106.0
C 209.0,94.0 206.0,72.0 201.0,57.0 C 193.0,36.0
184.0,30.0 163.0,31.0 C 147.0,31.0 129.0,34.0
123.0,37.0" fill="black" stroke="#000000" stroke-
width="0.1987360650449424"/>
</svg>
```

```
<svg baseProfile="full" height="600px" version="1.1"
viewBox="71.52534194356325 33.091703505641036
168.3279293439418 235.23462435726688"
width="430px">
  <defs/>
  <path d="M 166.0,66.0 C 163.0,75.0 160.0,94.0
160.0,110.0 C 160.0,130.0 154.0,141.0 139.0,145.0 C
110.0,155.0 105.0,145.0 117.0,105.0 C 123.0,85.0
124.0,68.0 119.0,65.0 C 115.0,62.0 108.0,72.0 104.0,87.0
C 100.0,102.0 94.0,119.0 90.0,125.0 C 76.0,143.0
98.0,180.0 123.0,180.0 C 144.0,180.0 146.0,184.0
142.0,216.0 C 139.0,244.0 141.0,251.0 151.0,248.0 C
159.0,245.0 164.0,235.0 163.0,224.0 C 161.0,197.0
178.0,170.0 199.0,170.0 C 210.0,170.0 220.0,163.0
224.0,155.0 C 227.0,147.0 226.0,141.0 222.0,141.0 C
184.0,147.0 177.0,140.0 183.0,100.0 C 188.0,59.0
177.0,37.0 166.0,66.0" fill="black" stroke="#000000"
stroke-width="0.19586563227083006"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="45.806005284816486
32.136951965994925 179.00915705709124
233.38494881101258" width="461px">
  <defs/>
  <path d="M 125.0,57.0 C 114.0,59.0 95.0,63.0
83.0,66.0 C 56.0,71.0 54.0,90.0 75.0,130.0 C
86.0,151.0 98.0,160.0 114.0,160.0 C 177.0,160.0
192.0,195.0 133.0,205.0 C 112.0,209.0 91.0,213.0
86.0,214.0 C 81.0,215.0 85.0,224.0 95.0,233.0 C
116.0,255.0 165.0,248.0 194.0,220.0 C 233.0,183.0
183.0,120.0 115.0,120.0 C 100.0,120.0 90.0,114.0
90.0,106.0 C 90.0,89.0 148.0,77.0 184.0,86.0 C
215.0,93.0 219.0,76.0 191.0,60.0 C 171.0,50.0
160.0,49.0 125.0,57.0" fill="black"
stroke="#000000" stroke-
width="0.19432551940966908"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="57.170055078209444
10.267723163708588 175.06296916140207
236.87755403778914" width="444px">
  <defs/>
  <path d="M 138.0,77.0 C 76.0,164.0 64.0,189.0
76.0,208.0 C 91.0,232.0 130.0,234.0 165.0,212.0 C
220.0,178.0 237.0,110.0 191.0,110.0 C 170.0,110.0
120.0,163.0 120.0,186.0 C 120.0,194.0 116.0,200.0
110.0,200.0 C 93.0,200.0 99.0,180.0 127.0,142.0 C
143.0,122.0 158.0,103.0 161.0,100.0 C 171.0,92.0
190.0,52.0 190.0,41.0 C 190.0,18.0 171.0,32.0
138.0,77.0 M 190.0,147.0 C 190.0,157.0
158.0,190.0 148.0,190.0 C 143.0,190.0 147.0,179.0
156.0,165.0 C 171.0,142.0 190.0,132.0
190.0,147.0" fill="black" stroke="#000000" stroke-
width="0.19723360036452053"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1"
viewBox="78.70432159914272
54.61954263571997 134.56958353012251
235.0494586955932" width="344px">
  <defs/>
  <path d="M 90.0,87.0 C 90.0,99.0
111.0,105.0 134.0,101.0 C 139.0,100.0
149.0,101.0 157.0,104.0 C 167.0,108.0
162.0,126.0 135.0,178.0 C 100.0,247.0
93.0,270.0 108.0,270.0 C 127.0,270.0
212.0,99.0 201.0,83.0 C 191.0,69.0 90.0,73.0
90.0,87.0" fill="black" stroke="#000000"
stroke-width="0.19571145603296686"/>
</svg>
```

```
<svg baseProfile="full" height="600px" version="1.1"
viewBox="53.46091743629355 36.534792829960466
191.74591129561253 227.73555388984175"
width="506px">
  <defs/>
  <path d="M 108.0,63.0 C 77.0,75.0 73.0,94.0
92.0,135.0 C 102.0,158.0 102.0,163.0 87.0,171.0 C
60.0,187.0 65.0,229.0 97.0,241.0 C 146.0,259.0
209.0,216.0 194.0,175.0 C 190.0,165.0 195.0,153.0
209.0,144.0 C 269.0,101.0 183.0,33.0 108.0,63.0 M
190.0,109.0 C 190.0,119.0 150.0,131.0 138.0,124.0 C
122.0,114.0 131.0,104.0 160.0,102.0 C 177.0,101.0
190.0,104.0 190.0,109.0 M 155.0,189.0 C 162.0,202.0
123.0,223.0 109.0,215.0 C 102.0,210.0 103.0,204.0
111.0,194.0 C 124.0,178.0 146.0,176.0 155.0,189.0"
fill="black" stroke="#000000" stroke-
width="0.1896216102330073"/>
</svg>
```

```
<svg baseProfile="full" height="600px"
version="1.1" viewBox="65.4721936264042
40.58183152173913 167.2965978728934
239.45590217391305" width="420px">
  <defs/>
  <path d="M 122.0,81.0 C 66.0,117.0 65.0,180.0
121.0,180.0 C 145.0,180.0 151.0,183.0 146.0,196.0 C
129.0,241.0 128.0,260.0 144.0,260.0 C 153.0,260.0
160.0,252.0 160.0,242.0 C 160.0,232.0 171.0,202.0
184.0,177.0 C 208.0,128.0 201.0,100.0 174.0,139.0 C
164.0,152.0 148.0,160.0 129.0,160.0 C 91.0,160.0
91.0,139.0 131.0,106.0 C 164.0,78.0 196.0,72.0
203.0,92.0 C 205.0,99.0 211.0,102.0 215.0,98.0 C
219.0,94.0 220.0,84.0 217.0,75.0 C 208.0,54.0
161.0,56.0 122.0,81.0" fill="black"
stroke="#000000" stroke-
width="0.1993804347826087"/>
</svg>
```

Figure 13: Examples of our SGP-MNIST challenge, hand-written digit constructed by SVG programs.

SGP-MNIST. The MNIST SVG data is sampled from the kaggle *MNIST-SVG* dataset[6]. We randomly sample 1000 samples from *MNIST-SVG* (100 samples per digit category) to build our SPG-MNIST benchmark. The data comes with CC BY-SA 3.0 license.

[1] https://github.com/lupantech/MathVista

[2] https://github.com/vllm-project/vllm

[3] https://github.com/openai/simple-evals

[4] https://www.kaggle.com/datasets/victorcondino/svgicons

[5] https://www.svgrepo.com/page/licensing/

[6] https://www.kaggle.com/datasets/jacekpardyak/mnist-svg

A.2 Evaluation protocol

Inference. To conduct massive-scale evaluation (8000+ question samples with 16 models), we leverage vLLM[7] to perform high-throughput and memory-efficient inference for all the open-source LLMs. And we use OpenAI API for evaluating different variants of the GPT models. We deploy the vLLM inference engine as a server that also uses the same format as OpenAI API to achieve a unified testing framework for both GPT and all open-source models. The vLLM inference engine is deployed on a node with 8 NVIDIA H100 80G GPUs.

Evaluation. We benchmark the performance of all the models via the question answering accuracy. Following the common protocol [34], we ask the model to generate the answer sentence in a formatted way (see text prompt examples in F.2). Then, we extract the target answer from the output sentence by parsing the answer according to its position. If the extracted answer matches the ground truth, it will count as 1, otherwise it's 0. We use the average accuracies for the results in Table 1.

Enhanced answer extraction with LLM: In our experiment, we found the Symbolic Instruction Tuning makes the model less capable in following the formatting instruction. This is likely due to our fine-tuning only uses symbolic graphics description, which causes the model to forget its instruction following skill. Therefore, the model after SIT often answers questions correctly but in a different format. This affects the aforementioned format-based answer extraction. For example, given a color-grounding question of the input subject, the formatted answer should be "The answer is A) Yellow.", however, the model outputs "The car is yellow". To mitigate this issue, we follow the GPT-enhanced answer extraction of Mathvista[8], where we present both the question options and model's output to GPT4 to extract the answer in the formatted way. A 5-shot CoT is also applied here to augment the robustness of extraction process (More details in F.2). The results on Table 2 are obtained with the enhanced answer extraction. More details of the SIT evaluation can be found in Appendix E.

A.3 Evaluated model specs

Here we list the model details of all LLMs we evaluated in the SGP-Bench, their performance are demonstrated in the table 2. Generally, we evaluated 3 types of LLMs, the representative open-sourced LLMs from tech giants and start-ups, the code-specific LLMs that were built for code generation and understanding and the strongest proprietary models from the GPT and Claude family.

A.3.1 Open-sourced LLMs

Gemma-1.1-2B/7B Gemma is a suite of lightweight, advanced open models created by Google DeepMind and other teams across Google, it's the best performing model at it class when it's released on Feb 21, 2024. Primarily designed for text generation, Gemma models come in multiple sizes, i.e. 2B / 7B, to fit various computing resources and deployment needs. The models are trained on 3T (2B) / 6T (7B) tokens of primarily-English data from web, mathematics, and code. It is based on a transformer decoder with context length of 8192 tokens. It leverages Multi-Query Attention, RoPE Embeddings, GeGLU Activations and RMSNorm. The Gemma models are using architectures. data and training recipes inspired by the Gemini model family. The models are available in two versions: a pretrained version and an Instruction-tuned version, the latter refined through human language interactions to perform well in conversational roles, similar to a chat bot. We only test and perform the symbolic instruction tuning on the Instruction tuned version.

Mistral-0.3-7B The Mistral-0.1-7B from Mistral AI is released September 27, 2023 and marked as the best 7B model at the time. The 0.1 version model features a 8k context window, with Grouped-query attention (GQA) for faster inference and SWA for handling longer sequences more effectively at a reduced computational cost. The model is further updated to 0.3 version in May 21, 2024, upgrading its context length to 32k, its vocabulery size and RoPE theta, but the SWA is removed in this version.

Mistral-NeMo and Mistral-Large2 Mistral NeMo is a 12B large language model built by Mistral AI with a context window size of up to 128k tokens. Mistral NeMo is trained with quantization

[7] https://github.com/vllm-project/vllm

[8] https://github.com/lupantech/MathVista

awareness, allowing FP8 inference without any loss in performance. Mistral NeMo uses a new tokenizer, Tekken, based on Tiktoken, which enables more efficient compression of natural language text and source code, compared with previous Mistral model series.

Mistral Large 2 is the new generation of Mistral AI's flagship model, with a model size of 123 billion parameters. Especially, Mistral Large 2 is trained on a very large proportion of code data, resulting in state-of-the-art performance, on par with proprietary models like GPT-4o or Clause Opus.

Yi-1.5-9B/34B The Yi model family, developed by LLM-focused startup 01.AI, includes 6B and 34B pretrained language models. Their performance is attributed to high-quality data from meticulous data-engineering efforts. For pretraining, 3.1 trillion tokens of English and Chinese corpora were constructed using a cascaded data deduplication and quality filtering pipeline. Finetuning involved a carefully refined instruction dataset of fewer than 10K instances, each verified by dedicated machine learning engineers. Built on Transformer architecture, Yi models feature Grouped-Query Attention (GQA), SwiGLU activation, and RoPE with an adjusted base frequency (RoPE ABF). The Yi-6B base model, with 32 layers, was scaled up to the Yi-9B model, which has 48 layers, by duplicating the original 16 middle layers (layers 12-28). We hence test the Yi-9B model together with the 34B version in SGP-Bench.

InternLM2-7B/InternLM2-20B/InternLM2.5-7B InternLM2 is a open-sourced LLM model series developed by Shanghai AI laboratory, with a context window length of 200K.

InternLM2.5 is an open-sourced, 7 billion parameter base and chat model with a context window size of 1M. It supports gathering information from more than 100 web pages and has in general a very strong capability in tool utilization.

Aya-23-8B/35B Aya 23 is an open-source LLM model series developed by C4AI, featuring advanced multilingual capabilities. Aya-23 is fine-tuned (IFT) to follow human instructions and supports a context window length of 8192 tokens.

Command R-35B/104B C4AI Command-R is a research release of a 35B large language model with open weights, optimized for a variety of use cases including reasoning, summarization, and question answering. Command-R has the capability for multilingual generation evaluated in 10 languages and highly performant RAG capabilities. It supports a context length of 128K.

C4AI Command-R+ is an open-source multilingual LLM with enhanced features, including Retrieval Augmented Generation (RAG) and tool usage for automating complex tasks. Command-R+ excels in multi-step tool usage, allowing the model to combine various tools across multiple steps to complete sophisticated tasks.

Qwen-1.5-7B/32B/72B/110B Qwen, released in April 2024, developed by Alibaba Cloud, is a series of transformer-based large language models pre-trained on diverse data, including web texts, books, code, and more, over 2.2 trillion tokens. The Qwen1.5 series includes various sizes of decoder models, each available as a base and aligned chat model, supporting long context lengths (8K tokens for 1.8B, 7B, and 14B models, and 32K tokens for the 72B model). It outperforms similar-scale open-source models on various Chinese and English tasks and even exceeds some larger models in benchmarks. These models feature the Transformer architecture with SwiGLU activation, attention QKV bias, group query attention, and a mix of sliding window and full attention mechanisms. They also include an advanced tokenizer for multiple languages and coding languages. Qwen's extensive vocabulary of over 150K tokens enhances compatibility with multiple languages, allowing for improved capabilities without expanding the vocabulary.

Qwen-2-72B Qwen2, is the newest series of large language models, developed by Alibaba, which surpasses its previous released Qwen1.5 series, yielding state-of-the-art performance across different benchmarks. Qwen2-72B-Instruct has an extended context length of up to 128K and is instruction-aligned with both supervised finetuning and direct preference optimization.

Llama3-8B/70B Meta's Llama 3 is the latest generation of llama family, release in April 18, 2024, featuring pretrained and instruction-fine-tuned versions with 8B and 70B parameters. Designed with a standard decoder-only transformer architecture, Llama 3 models demonstrate state-of-the-art performance across various industry benchmarks and show improved reasoning capabilities. Key enhancements include a tokenizer with a 128K token vocabulary for efficient language encoding and grouped query attention (GQA) for better inference efficiency.

Llama 3 models are pretrained on an extensive dataset of over 15T tokens from publicly available sources, including a significant increase in code and high-quality non-English data covering 30+ languages. This dataset is seven times larger than that used for Llama 2, ensuring superior model performance.

For instruction-tuning, Llama 3 employs a combination of supervised fine-tuning (SFT), rejection sampling, proximal policy optimization (PPO), and direct preference optimization (DPO). This approach, coupled with meticulously curated data and multiple quality assurance rounds, significantly enhances model alignment and response diversity.

Llama3.1-8B/70B/405B Introduced in July 2024, Llama 3.1 was pretrained on 15 trillion tokens of data from publicly available sources as well as over 25M synthetically generated examples. The intruction-tuned variants are post-trained using supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF) to align with human preferences to guarantee safety and helpfulness. The Llama 3.1 405B demonstrates competitive performance across 150 benchmarks with leading foundation models, including GPT-4o and Claude 3.5 Sonnet.

A.3.2 CODE-SPECIFIC LLMS

CodeQwen1.5-7B CodeQwen1.5-7B is based on Qwen1.5-7B. It is further trained on 3T tokens of code data, and it also includes group query attention (GQA) for efficient inference.

DeepSeek-Coder-V2-16B-Instruct DeepSeek-Coder-V2-16B-Instruct is an Mixture-of-Experts (MoE) code language model, that demonstrates comparable performance to GPT-4 Turbo in code-related tasks. Specifically, DeepSeek-Coder-V2 is continued to pre-train on intermediate checkpoint of DeepSeek-V2 with 6 trillion additional tokens, substantially enhance its reasoning capabilities in code and mathematical related tasks. DeepSeek-Coder-V2 supports 338 programming languages and has a context length of 128K.

Codestral-22B-v0.1 Codestral-22B-v0.1 is the code-specific variant of Mistral-0.1-22B, it's trained on a diverse dataset of 80+ programming languages, including the most popular ones, such as Python, Java, C, C++, JavaScript, and Bash.

A.3.3 GPT FAMILY

GPT-3.5t (GPT-3.5 Turbo) is a text only language model released by OpenAI on November 2022. The specific version of the model we are using is *gpt-3.5-turbo-0125*. It has a knowledge cutoff of September 2021 and a context window with length of 16K tokens.

GPT-4t (GPT-4 Turbo) is vision language model launched by OpenAI on March 2023. The specific version of the model we are using is *gpt-4-turbo-2024-04-09*. It has an updated knowledge cutoff of April 2023 and a context window with length of 128K tokens. It is more powerful than GPT-3.5.

GPT-4o (GPT-4 Omni) is a multimodal model released by OpenAI on May 2024, which support data types such as audio, vision, and text. The specific version of the model we are using is *gpt-4o-2024-05-13*. It has similar performance as GPT-4t on English text and code, but with significant improvement on non-English text, *i.e.*, over 50 languages. At the same time, it is able to reason with vision input. GPT-4o has knowledge up to October 2023 and supports context window with length of 128K tokens.

GPT-4o mini is a multimodal model released by OpenAI in July 2024, which is a more cost-efficient and smaller modal than GPT-4. It has a context window size of 128K tokens and the has knowledge up to October 2023.

A.3.4 CLAUDE FAMILY

Claude is a multimodal, multilingual, proprietary model series developed by Anthropic. The Claude series includes different models: **Haiku**, the fastest and most lightweight model; **Sonnet**, the best balanced model between performance and speed; and **Opus**, the highest-performing model. We did not evaluate **Claude 3 Opus** because, in June 2024, Anthropic released **Claude 3.5 Sonnet**, the newest best-performing model.

Specifically, we are using *claude-3-5-sonnet-20240620* for **Claude 3.5 Sonnet**, *claude-3-sonnet-20240229* for **Claude 3 Sonnet**, and *claude-3-haiku-20240307* for **Claude 3 Haiku** for benchmark evaluation.

B Human Study Details

We ran a human study to verify the labels produced by GPT-4o for the benchmark over a subset of 500 stimuli. We recruited 55 participants from the crowdsourcing platform, Prolific [73]. Stimuli were batched into 10 sets of 50 stimuli each. Each participant was randomly assigned a batch of 50 stimuli; stimuli were presented in a random shuffled. On each trial, participants saw the question, original image, and set of multiple choice options. Participants selected an option by clicking a button. We include an example screenshot of a trial in Figure 14. Participants were paid at a base rate of $12.50/hr. They were informed that they could receive a bonus up to $\$15/hr$ based on the amount of correct answers they achieved. All participants received the full bonus. Our study was approved by our institutional ethics review board, and all participants provided informed consent. We include the set of instructions and sample screenshots in Figures 15 and 16, respectively. We found high inter-annotator agreement (participants in the same batch had between $0.7 - 0.85$ Fleiss Kappa's alpha agreement, where higher implies higher agreement). We find that humans' mode response matched GPT-4o on 90% of the examples (450 of the 500 stimuli).

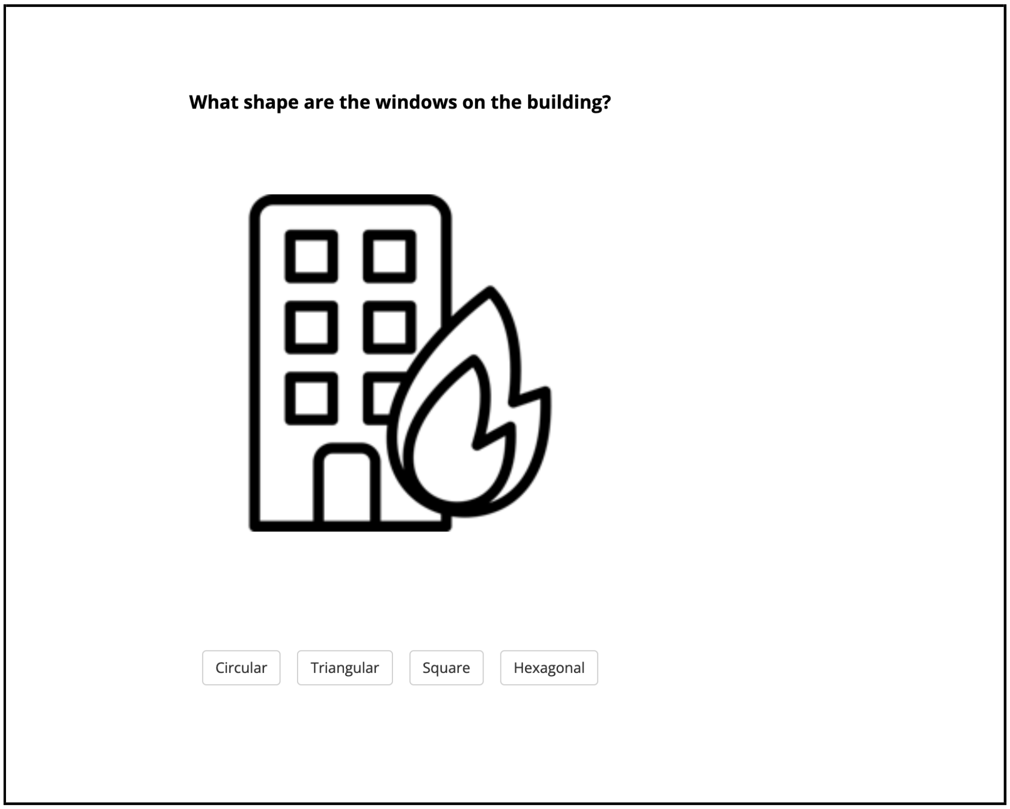
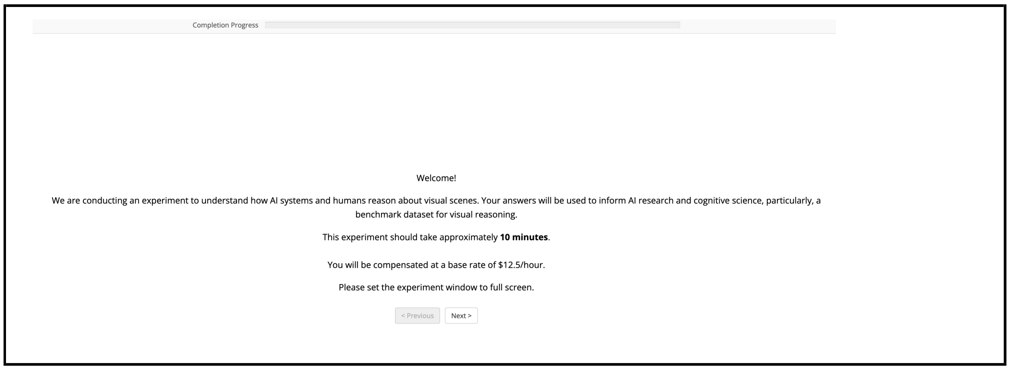

What shape are the windows on the building?

Figure 14: Example survey question.



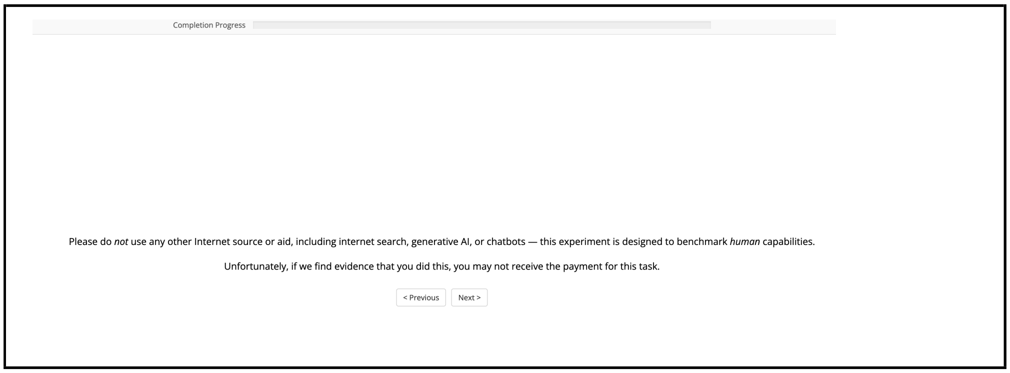
Welcome!

We are conducting an experiment to understand how AI systems and humans reason about visual scenes. Your answers will be used to inform AI research and cognitive science, particularly, a benchmark dataset for visual reasoning.

This experiment should take approximately **10 minutes**.

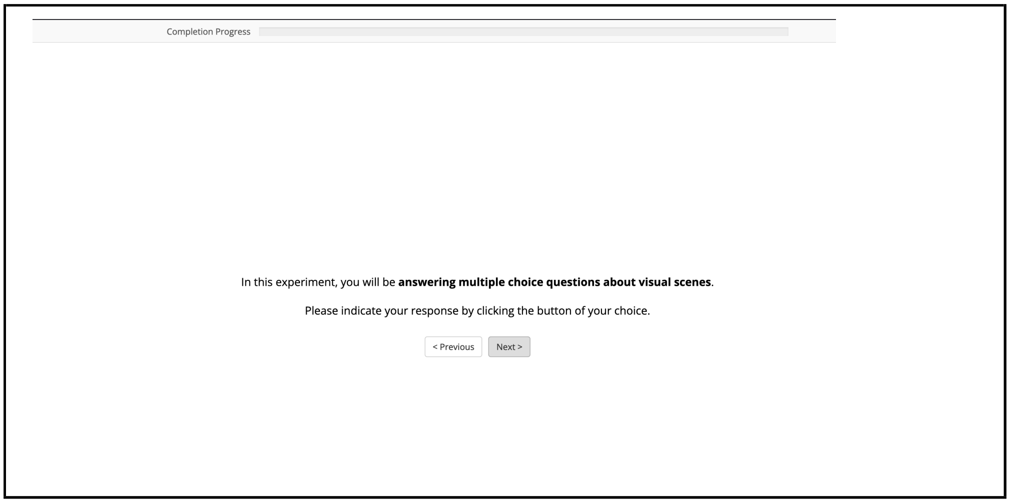
You will be compensated at a base rate of $12.5/hour.

Please set the experiment window to full screen.



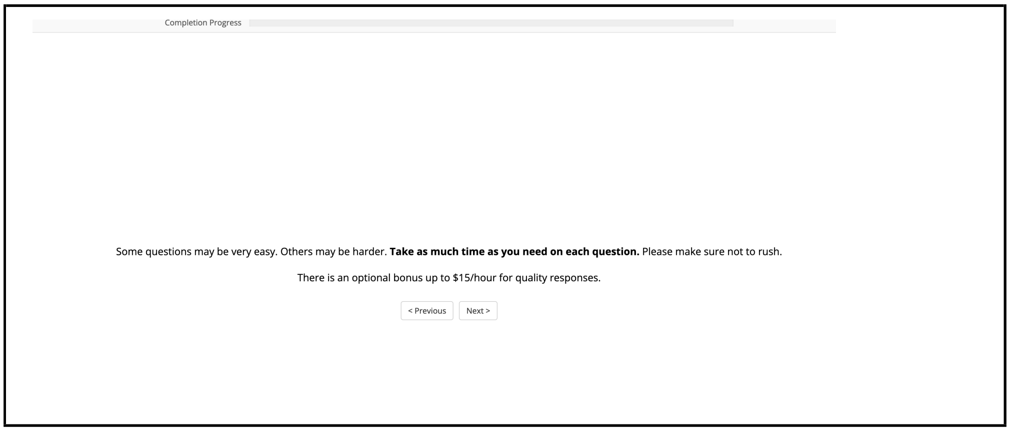
Please do *not* use any other Internet source or aid, including internet search, generative AI, or chatbots — this experiment is designed to benchmark *human* capabilities.

Unfortunately, if we find evidence that you did this, you may not receive the payment for this task.



In this experiment, you will be **answering multiple choice questions about visual scenes**.

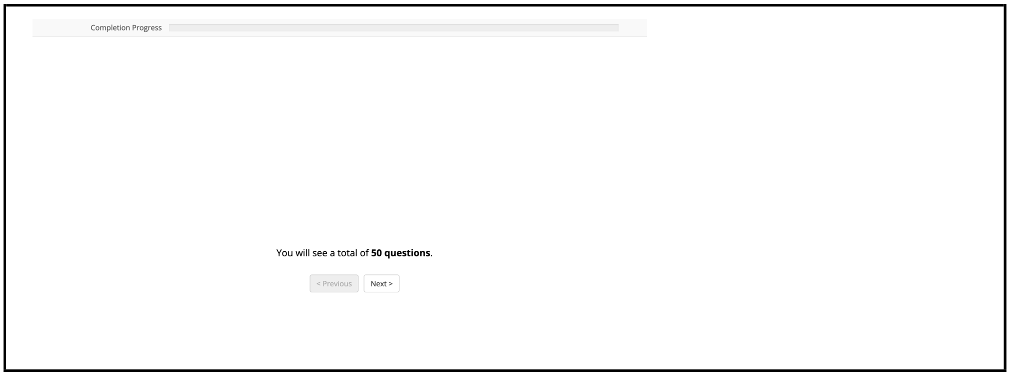
Please indicate your response by clicking the button of your choice.



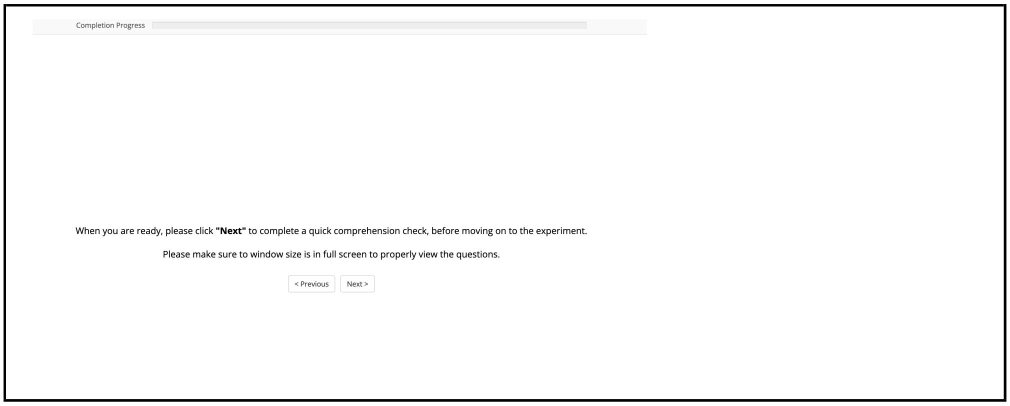
Some questions may be very easy. Others may be harder. **Take as much time as you need on each question.** Please make sure not to rush.

There is an optional bonus up to $15/hour for quality responses.

Figure 15: Experiment instructions.

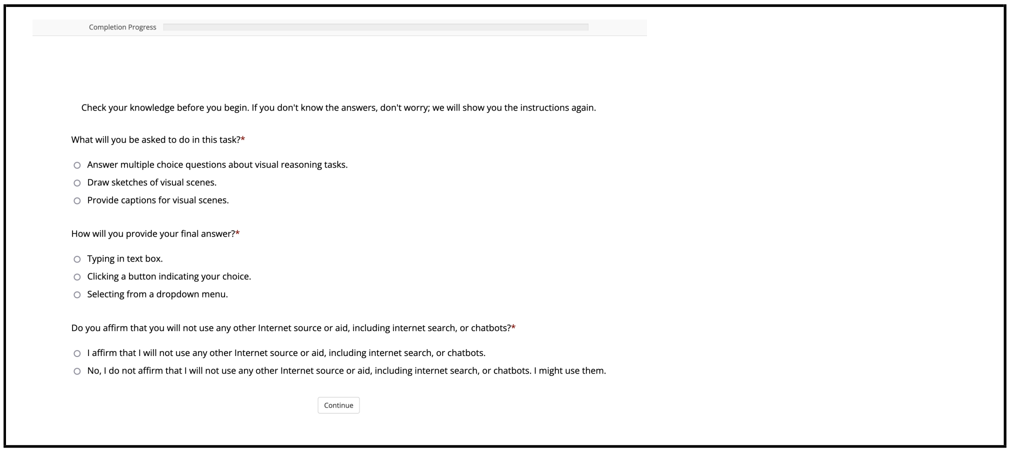


You will see a total of **50 questions**.



When you are ready, please click **"Next"** to complete a quick comprehension check, before moving on to the experiment.

Please make sure to window size is in full screen to properly view the questions.



Check your knowledge before you begin. If you don't know the answers, don't worry; we will show you the instructions again.

What will you be asked to do in this task?*

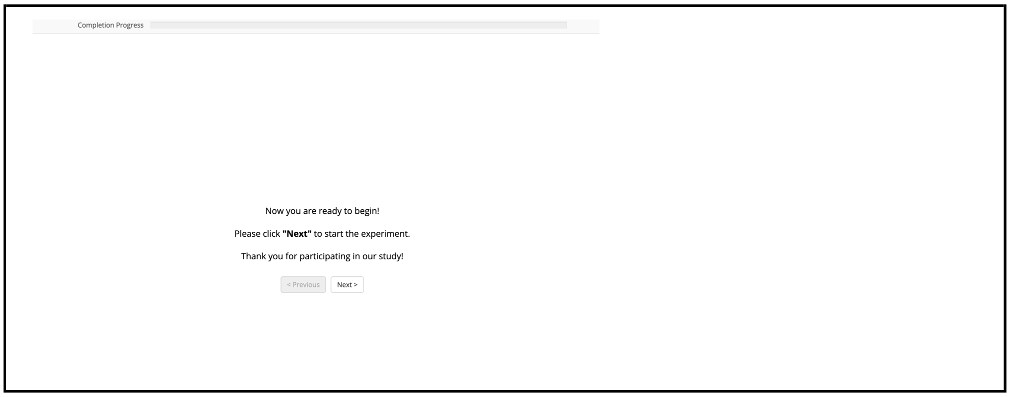
○ Answer multiple choice questions about visual reasoning tasks.
○ Draw sketches of visual scenes.
○ Provide captions for visual scenes.

How will you provide your final answer?*

○ Typing in text box.
○ Clicking a button indicating your choice.
○ Selecting from a dropdown menu.

Do you affirm that you will not use any other Internet source or aid, including internet search, or chatbots?*

○ I affirm that I will not use any other Internet source or aid, including internet search, or chatbots.
○ No, I do not affirm that I will not use any other Internet source or aid, including internet search, or chatbots. I might use them.



Now you are ready to begin!

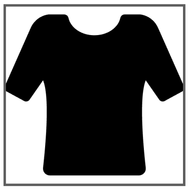
Please click **"Next"** to start the experiment.

Thank you for participating in our study!

Figure 16: Experiment instructions (continued).

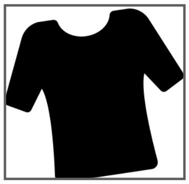
C SVG - Invariance Illustration

Our SVG - Invariance test is essential for testing whether a model has a fundamental understanding of the code, or it is able to pass the benchmark tests due to memorizing the SVG code samples, since we built our SVG-Bench using public available SVG datasets. In Figure 17 we see two SVG codes, illustrating two samples, that are semantically identical. The rotated sample is generated by ourself by applying a SE(2) transformation on the original sample (from *SVG Icons*). We can see that semantically these two samples are identical, the code changed drastically.



Original

```
<svg id="Capa_1" style="enable-background:new 0 0 298.667 298.667;" version="1.1" viewBox="0 0 298.667
298.667" x="0px" y="0px" xml:space="preserve">
  <g>
   <path
d="M0.604,134.717c-1.483,3.342-0.15,7.264,3.063,9.01l28.862,15.682c3.141,1.707,7.063,0.779,9.106-2.154l22.
406-32.165
c11.774,28.66,3.631,113.167-0.035,145.359c-0.367,3.219,0.658,6.442,2.817,8.858c2.159,2.416,5.246,3.792,8.48
6,3.792h148.047
c3.238,0,6.32-1.382,8.477-3.796s3.193-5.637,2.827-8.854c-3.666-32.192-11.809-116.698-0.035-145.359l22.406,
32.165   c2.043,2.933,5.965,3.861,9.106,2.154L295,143.727c3.213-1.746,4.546-5.667,3.063-9.01L255.015,37.71
c-4.193-9.437-12.08-16.759-21.829-20.249c-2.582-0.924-5.233-1.548-7.908-1.892l-25.618,0c-3.516,0-6.541,2.45
1-7.28,5.889
c-3.519,16.365-21.452,28.822-43.046,28.822c-21.594,0-39.527-12.456-43.046-28.821c-0.738-3.431-3.771-5.889
-7.28-5.889l-25.618,0
c-2.674,0.344-5.326,0.969-7.908,1.892c-9.749,3.49-17.636,10.813-21.829,20.249L0.604,134.717z" />
  </g>
</svg>
```



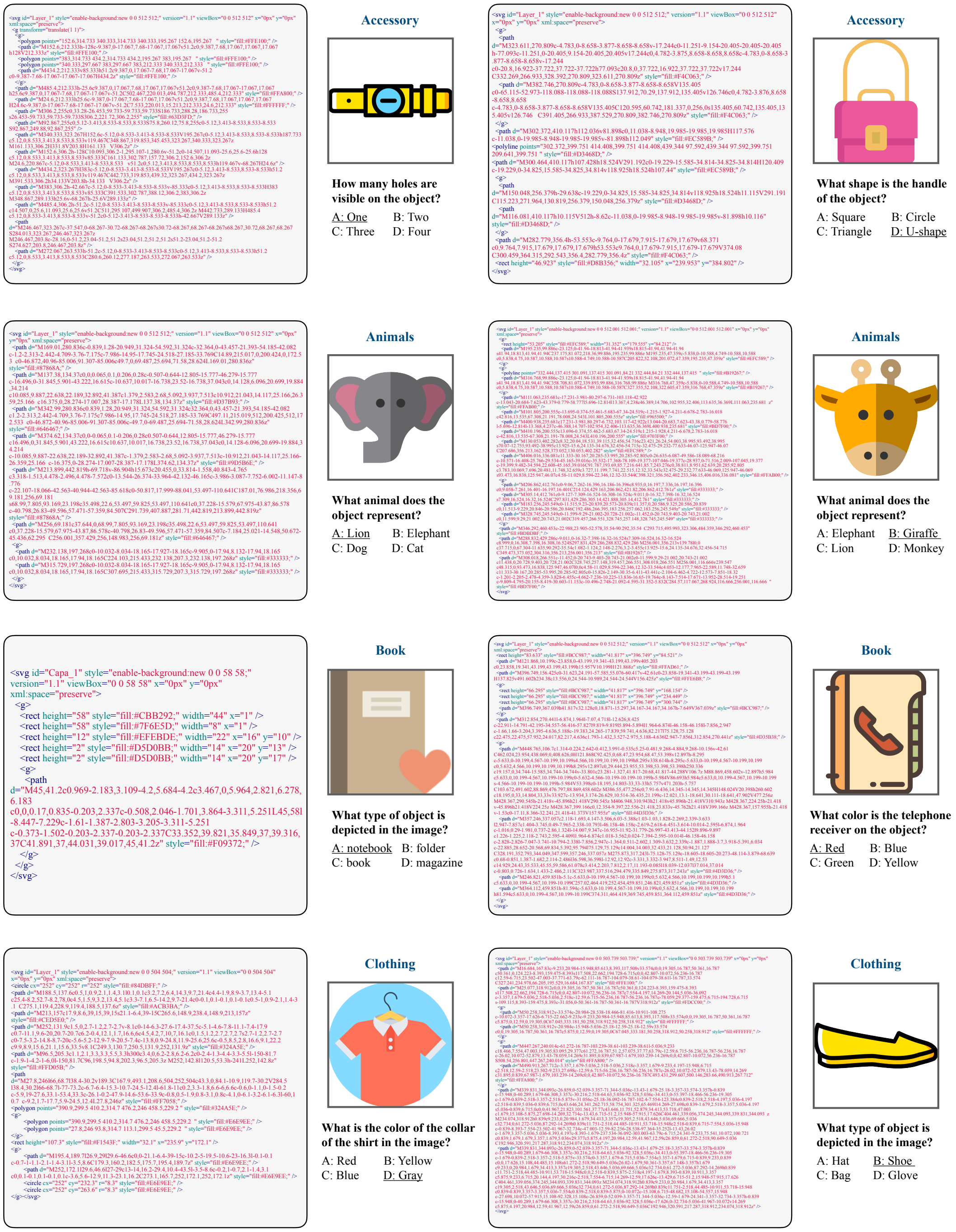
Rotated

```
<svg id="Capa_1" style="enable-background:new 0 0 298.667 298.667;" version="1.1" viewbox="0 0 298.667 298.667"
x="0px" xml:space="preserve" y="0px">
  <g>
   <path
d="M0.08497124091795172,157.01353292632658C-0.8841617683852974,160.53901646908758,1.017662534846778,164.
2189671915343,4.454721950394287,165.46736919970158L35.32910693826702,176.67952865364342C38.689164173223
62,177.9000799134501,42.42938657654092,176.39874571557027,44.0131506388647,173.19436911552907L61.38288566
434426,138.05310610258903C77.2909273289573,164.64176696388355,81.81484833500208,249.41956852025248,82.980
45135812532,281.7986634673397C83.09658635086157,285.03643471674286,84.58981594452071,288.0710054245401,8
7.08432092832022,290.13880180856813C89.57882591211973,292.2065981925961,92.83622538891045,293.1078715410
3343,96.04014623172823,292.62569809406875L242.43856259488945,270.59349676604694C245.64050570879192,270.1
116209570124,248.48251815751382,268.2863502260496,250.25624981090428,265.57822891751 1S252.57480151787246,
259.52882101407823,251.7341265156018,256.40211179501784C243.31816928891686,225.11415750013606,222.689750
5443896,142.7610104740483,230.06734030978453,112.6669717578251L257.0105993609562,141.1393593061403C259.4
673354726304,143.73566161473153,263.4837658751966,144.0696598502683,266.3357551632589,141.91422785162442
L292.5435709937984,122.11149363655903C295.4609545831728,119.90678095857828,296.1955917430269,115.8310677
3959912,294.23160395194606,112.74599238290436L237.22648542993093,23.225576748865834C231.67577218519597,1
4.517660782647283,222.78694574507878,8.450930401505076,212.62712795911045,6.450629535154889C209.93637120
566882,5.921169344197892,207.2220284941824,5.698636999131651,204.5256223459437,5.756558357372768L179.1928
9266961482,9.569002605626793C175.71604523648296,10.092250086962622,173.08948553555297,12.9661342421523 5,
172.8703543033994,16.475827462804972C171.82596264096188,33.18228825063885,155.94649417015597,48.16934330
1070896,134.59295507145015,51.38294003371189C113.2394159727443,54.5965367663529,93.65242062408873,44.9480
1158310621,87.73718423351656,29.288938671734655C86.49680439446237,26.005973113756937,83.13178147813346,2
4.026712198015574,79.6618560962052,24.54891794659561L54.3291264198763,28.36136219484962C51.7360965840926
9,29.099473480995627,49.20663989557427,30.11218166251159,46.790751770695564,31.409154124852236C37.6696903
61213,36.31112717296749,30.960317665169526,44.72631677425315,28.21826474900425,54.68123971744809L0.084971
24091795172,157.01353292632658z"/>
  </g>
</svg>
```

Figure 17: Illustration of the SVG - Invariance test.

D More Examples in SGP-Bench

D.1 SVG Data



```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <g transform="translate(1 1)">
   <g>
    <polygon points="152.6,314.733 340.333,314.733 340.333,195.267 152.6,195.267   " style="fill:#FFE100;" />
    <path d="M152.6,212.333h-128c-9.387,0-17.067,7.68-17.067,17.067v51.2c0,9.387,7.68,17.067,17.067,17.067
h128V212.333z" style="fill:#FFE100;" />
    <polygon points="383,314.733 434.2,314.733 434.2,195.267 383,195.267   " style="fill:#FFE100;" />
    <polygon points="340.333,297.667 383,297.667 383,212.333 340.333,212.333   " style="fill:#FFE100;" />
    <path d="M434.2,212.333v85.333h51.2c9.387,0,17.067-7.68,17.067-17.067v-51.2
c0-9.387-7.68-17.067-17.067-17.067H434.2z" style="fill:#FFE100;" />
   </g>
   <path d="M485.4,212.333h-25.6c9.387,0,17.067,7.68,17.067,17.067v51.2c0,9.387-7.68,17.067-17.067,17.067
h25.6c9.387,0,17.067-7.68,17.067-17.067v-51.2C502.467,220.013,494.787,212.333,485.4,212.333" style="fill:#FFA800;" />
   <path d="M24.6,212.333h25.6c-9.387,0-17.067,7.68-17.067,17.067v51.2c0,9.387,7.68,17.067,17.067,17.067
H24.6c-9.387,0-17.067-7.68-17.067-17.067v-51.2C7.533,220.013,15.213,212.333,24.6,212.333" style="fill:#FFFFFF;" />
   <path d="M306.2,255c0,33.28-26.453,59.733-59.733,59.733S186.733,288.28,186.733,255
s26.453-59.733,59.733-59.733S306.2,221.72,306.2,255" style="fill:#63D3FD;" />
   <path d="M92.867,255c0,5.12-3.413,8.533-8.533,8.533S75.8,260.12,75.8,255c0-5.12,3.413-8.533,8.533-8.533
S92.867,249.88,92.867,255" />
   <path d="M340.333,323.267H152.6c-5.12,0-8.533-3.413-8.533-8.533V195.267c0-5.12,3.413-8.533,8.533-8.533h187.733
c5.12,0,8.533,3.413,8.533,8.533v119.467C348.867,319.853,345.453,323.267,340.333,323.267z
M161.133,306.2H331.8V203.8H161.133   V306.2z" />
   <path d="M152.6,306.2h-128C10.093,306.2-1,295.107-1,280.6v-51.2c0-14.507,11.093-25.6,25.6-25.6h128
c5.12,0,8.533,3.413,8.533,8.533v85.333C161.133,302.787,157.72,306.2,152.6,306.2z
M24.6,220.867c-5.12,0-8.533,3.413-8.533,8.533   v51.2c0,5.12,3.413,8.533,8.533,8.533h119.467v-68.267H24.6z" />
   <path d="M434.2,323.267H383c-5.12,0-8.533-3.413-8.533-8.533V195.267c0-5.12,3.413-8.533,8.533-8.533h51.2
c5.12,0,8.533,3.413,8.533,8.533v119.467C442.733,319.853,439.32,323.267,434.2,323.267z
M391.533,306.2h34.133V203.8h-34.133   V306.2z" />
   <path d="M383,306.2h-42.667c-5.12,0-8.533-3.413-8.533-8.533v-85.333c0-5.12,3.413-8.533,8.533-8.533H383
c5.12,0,8.533,3.413,8.533,8.533v85.333C391.533,302.787,388.12,306.2,383,306.2z
M348.867,289.133h25.6v-68.267h-25.6V289.133z" />
   <path d="M485.4,306.2h-51.2c-5.12,0-8.533-3.413-8.533-8.533v-85.333c0-5.12,3.413-8.533,8.533-8.533h51.2
c14.507,0,25.6,11.093,25.6,25.6v51.2C511,295.107,499.907,306.2,485.4,306.2z M442.733,289.133H485.4
c5.12,0,8.533-3.413,8.533-8.533v-51.2c0-5.12-3.413-8.533-8.533-8.533h-42.667V289.133z" />
   <path
d="M246.467,323.267c-37.547,0-68.267-30.72-68.267-68.267s30.72-68.267,68.267-68.267s68.267,30.72,68.267,68.267
S284.013,323.267,246.467,323.267z
M246.467,203.8c-28.16,0-51.2,23.04-51.2,51.2s23.04,51.2,51.2,51.2s51.2-23.04,51.2-51.2
S274.627,203.8,246.467,203.8z" />
   <path d="M272.067,263.533h-51.2c-5.12,0-8.533-3.413-8.533-8.533c0-5.12,3.413-8.533,8.533-8.533h51.2
c5.12,0,8.533,3.413,8.533,8.533C280.6,260.12,277.187,263.533,272.067,263.533z" />
  </g>
</svg>
```

Accessory

How many holes are visible on the object?

A: One B: Two
C: Three D: Four

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512"
x="0px" y="0px" xml:space="preserve">
  <g>
   <path
d="M323.611,270.809c-4.783,0-8.658-3.877-8.658-8.658v-17.244c0-11.251-9.154-20.405-20.405-20.405
h-77.093c-11.251,0-20.405,9.154-20.405,20.405v17.244c0,4.782-3.875,8.658-8.658,8.658c-4.783,0-8.658-3
.877-8.658-8.658v-17.244
c0-20.8,16.922-37.722,37.722-37.722h77.093c20.8,0,37.722,16.922,37.722,37.722v17.244
C332.269,266.933,328.392,270.809,323.611,270.809z" style="fill:#F4C063;" />
   <path d="M382.746,270.809c-4.783,0-8.658-3.877-8.658-8.658V135.405
c0-65.115-52.973-118.088-118.088-118.088S137.912,70.29,137.912,135.405v126.746c0,4.782-3.876,8.658
-8.658,8.658
c-4.783,0-8.658-3.877-8.658-8.658V135.405C120.595,60.742,181.337,0,256,0s135.405,60.742,135.405,13
5.405v126.746   C391.405,266.933,387.529,270.809,382.746,270.809z" style="fill:#F4C063;" />
  </g>
  <path d="M302.372,410.117h112.036v81.898c0,11.038-8.948,19.985-19.985,19.985H117.576
c-11.038,0-19.985-8.948-19.985-19.985v-81.898h112.049" style="fill:#EC589B;" />
<polyline points="302.372,399.751 414.408,399.751 414.408,439.344 97.592,439.344 97.592,399.751
209.641,399.751 " style="fill:#D3468D;" />
<path d="M300.464,410.117h107.428h18.524V291.192c0-19.229-15.585-34.814-34.825-34.814H120.409
c-19.229,0-34.825,15.585-34.825,34.814v118.925h18.524h107.44" style="fill:#EC589B;" />
  <g>
   <path
d="M150.048,256.379h-29.638c-19.229,0-34.825,15.585-34.825,34.814v118.925h18.524h11.115V291.191
C115.223,271.964,130.819,256.379,150.048,256.379z" style="fill:#D3468D;" />
   <path
d="M116.081,410.117h10.115V512h-8.62c-11.038,0-19.985-8.948-19.985-19.985v-81.898h10.116"
style="fill:#D3468D;" />
  </g>
  <path d="M282.779,356.4h-53.553c-9.764,0-17.679,7.915-17.679,17.679v68.371
c0,9.764,7.915,17.679,17.679,17.679h53.553c9.764,0,17.679-7.915,17.679-17.679V374.08
C300.459,364.315,292.543,356.4,282.779,356.4z" style="fill:#F4C063;" />
  <rect height="46.923" style="fill:#D8B356;" width="32.105" x="239.953" y="384.802" />
</svg>
```

Accessory

What shape is the handle of the object?

A: Square B: Circle
C: Triangle D: U-shape

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px"
y="0px" xml:space="preserve">
  <path d="M169.01,280.836c-0.839,1.28-20.949,31.324-54.592,31.324c-32.364,0-43.457-21.393-54.185-42.082
c-1.2-2.313-2.442-4.709-3.76-7.175c-7.986-14.95-17.745-24.518-27.185-33.769C14.89,215.017,0,200.424,0,172.5
3  c0-46.872,40.96-85.006,91.307-85.006c49.7,0,69.487,25.694,71.58,28.624L169.01,280.836z"
style="fill:#87868A;" />
  <path d="M137.38,134.37c0,0,0.065,0.1,0.206,0.28c-0.507-0.644-12.805-15.777-46.279-15.777
c-16.496,0-31.845,5.901-43.222,16.615c-10.637,10.017-16.738,23.52-16.738,37.043c0,14.128,6.096,20.699,19.884
,34.214
c10.085,9.887,22.638,22.189,32.892,41.387c1.379,2.583,2.68,5.092,3.937,7.513c10.912,21.043,14.117,25.166,26.3
59,25.166  c16.375,0,28.274-17.007,28.387-17.178L137.38,134.37z" style="fill:#D37B93;" />
  <path d="M342.99,280.836c0.839,1.28,20.949,31.324,54.592,31.324c32.364,0,43.457-21.393,54.185-42.082
c1.2-2.313,2.442-4.709,3.76-7.175c7.986-14.95,17.745-24.518,27.185-33.769C497.11,215.019,512,200.425,512,17
2.533  c0-46.872-40.96-85.006-91.307-85.006c-49.7,0-69.487,25.694-71.58,28.624L342.99,280.836z"
style="fill:#646467;" />
  <path d="M374.62,134.37c0,0-0.065,0.1-0.206,0.28c0.507-0.644,12.805-15.777,46.279-15.777
c16.496,0,31.845,5.901,43.222,16.615c10.637,10.017,16.738,23.52,16.738,37.043c0,14.128-6.096,20.699-19.884,3
4.214
c-10.085,9.887-22.638,22.189-32.892,41.387c-1.379,2.583-2.68,5.092-3.937,7.513c-10.912,21.043-14.117,25.166-
26.359,25.166  c-16.375,0-28.274-17.007-28.387-17.178L374.62,134.37z" style="fill:#9D5B6E;" />
  <path d="M213.899,442.819h-69.718v-86.904h15.673c20.455,0,33.814-1.558,40.843-4.765
c3.318-1.513,4.478-2.496,4.478-7.572c0-13.544-26.374-33.964-42.132-46.165c-3.986-3.087-7.752-6.002-11.147-8
.776
c-22.107-18.066-42.563-40.944-42.563-85.618c0-50.817,17.999-88.041,53.497-110.641C187.01,76.986,218.356,6
9.181,256,69.181
s68.99,7.805,93.169,23.198c35.498,22.6,53.497,59.825,53.497,110.641c0,37.228-15.579,67.975-43.87,86.578
c-40.798,26.83-49.596,57.471-57.359,84.507C291.739,407.887,281.71,442.819,213.899,442.819z"
style="fill:#87868A;" />
  <path d="M256,69.181c37.644,0,68.99,7.805,93.169,23.198c35.498,22.6,53.497,59.825,53.497,110.641
c0,37.228-15.579,67.975-43.87,86.578c-40.798,26.83-49.596,57.471-57.359,84.507c-7.184,25.021-14.548,50.672-
45.436,62.295  C256.001,357.429,256,148.983,256,69.181z" style="fill:#646467;" />
  <g>
   <path d="M232.138,197.268c0-10.032-8.034-18.165-17.927-18.165c-9.905,0-17.94,8.132-17.94,18.165
c0,10.032,8.034,18.165,17.94,18.165C224.103,215.433,232.138,207.3,232.138,197.268z" style="fill:#333333;" />
   <path d="M315.729,197.268c0-10.032-8.034-18.165-17.927-18.165c-9.905,0-17.94,8.132-17.94,18.165
c0,10.032,8.034,18.165,17.94,18.165C307.695,215.433,315.729,207.3,315.729,197.268z" style="fill:#333333;" />
  </g>
</svg>
```

Animals

What animal does the object represent?

A: Lion B: Elephant
C: Dog D: Cat

```
<svg id="Layer_1" style="enable-background:new 0 0 512.001 512.001;" version="1.1" viewBox="0 0 512.001 512.001" x="0px" y="0px"
xml:space="preserve">
  <g>
   <rect height="53.205" style="fill:#EFC589;" width="31.352" x="179.555" y="84.212" />
   <path d="M195.235,99.886c-23.125,0-41.94-18.813-41.94-41.939s18.815-41.94,41.94-41.94
s41.94,18.813,41.94,41.94C237.175,81.072,218.36,99.886,195.235,99.886z M195.235,47.359c-5.838,0-10.588,4.749-10.588,10.588
c0,5.838,4.75,10.587,10.588,10.587s10.588-4.749,10.588-10.587C205.822,52.108,201.072,47.359,195.235,47.359z" style="fill:#EFC589;" />
  </g>
  <g>
   <polyline points="332.444,137.415 301.091,137.415 301.091,84.21 332.444,84.21 332.444,137.415  " style="fill:#B19267;" />
   <path d="M316.768,99.886c-23.125,0-41.94-18.813-41.94-41.939s18.815-41.94,41.94-41.94
s41.94,18.813,41.94,41.94C358.708,81.072,339.893,99.886,316.768,99.886z M316.768,47.359c-5.838,0-10.588,4.749-10.588,10.588
c0,5.838,4.75,10.587,10.588,10.587s10.588-4.749,10.588-10.587C327.355,52.108,322.605,47.359,316.768,47.359z" style="fill:#B19267;" />
  </g>
   <path d="M111.063,235.681c-17.231-3.981-80.297-6.731-103.118-42.922
c-13.043-20.684-7.623-43.379-0.779-58.777l5.696-12.814l13.367,4.238c46.389,14.706,102.955,32.406,113.635,36.369L111.063,235.681  z"
style="fill:#FFAB00;" />
   <path d="M101.805,200.555c-13.695-0.374-55.461-5.683-67.34-24.519c-1.215-1.927-4.211-6.678-2.783-16.018
c42.816,13.535,67.308,21.191,78.008,24.543L101.805,200.555z" style="fill:#965500;" />
   <path d="M400.938,235.681c17.231-3.981,80.297-6.732,103.117-42.922c13.044-20.683,7.623-43.38,0.779-58.776
l-5.696-12.814l-13.368,4.237c-46.388,14.707-102.954,32.406-113.635,36.369L400.938,235.681" style="fill:#BD7F00;" />
   <path d="M410.196,200.555c13.696-0.374,55.462-5.683,67.34-24.519c1.215-1.928,4.211-6.678,2.783-16.018
c-42.816,13.535-67.308,21.191-78.008,24.543L410.196,200.555" style="fill:#703F00;" />
   <path d="M130.053,402.282c8.32,20.04,18.531,39.115,32.456,54.716c23.421,26.24,54.003,38.995,93.492,38.995
s70.07-12.755,93.492-38.995c13.925-15.6,24.135-34.676,32.456-54.715c-32.475-29.232-77.633-46.07-125.947-46.07
C207.686,356.213,162.528,373.052,130.053,402.282" style="fill:#EFC589;" />
   <path d="M406.016,336.081c11.333-30.167,20.285-53.995,20.285-92.805c0-26.635-6.087-49.586-18.089-68.216
c-10.571-16.408-25.766-29.534-45.165-39.016c-35.532-17.368-78.109-19.377-107.046-19.377c-28.937,0-71.516,2.009-107.045,19.377
c-19.399,9.482-34.594,22.608-45.165,39.016C91.787,193.69,85.7,216.641,85.7,243.276c0,38.811,8.951,62.639,20.285,92.805
c3.783,10.069,7.696,20.481,11.748,32.659c3.727,11.199,7.741,22.515,12.32,33.543c32.475-29.232,77.633-46.069,125.947-46.069
s93.473,16.838,125.947,46.07c4.58-11.029,8.594-22.346,12.32-33.544C398.321,356.562,402.233,346.15,406.016,336.081" style="fill:#FFAB00;" />
  <g>
   <path d="M206.862,412.761c0-9.06,7.262-16.396,16.186-16.396c8.935,0,16.197,7.336,16.197,16.396
c0,9.058-7.261,16.401-16.197,16.401C214.124,429.163,206.862,421.82,206.862,412.761z" style="fill:#333333;" />
   <path d="M305.14,412.761c0-9.127-7.309-16.524-16.308-16.524c-9.011,0-16.32,7.398-16.32,16.524
s7.309,16.524,16.32,16.524C297.831,429.286,305.14,421.888,305.14,412.761" style="fill:#333333;" />
   <path d="M183.256,245.549c0-11.515,9.23-20.839,20.573-20.839c11.357,0,20.586,9.325,20.586,20.839
c0,11.513-9.229,20.846-20.586,20.846C192.486,266.395,183.256,257.062,183.256,245.549z" style="fill:#333333;" />
   <path d="M328.745,245.549c0-11.599-9.29-21.002-20.728-21.002c-11.452,0-20.743,9.403-20.743,21.002
c0,11.599,9.29,21.002,20.743,21.002C319.457,266.551,328.745,257.148,328.745,245.549" style="fill:#333333;" />
  </g>
  <path d="M346.292,460.453c-22.988,23.905-52.578,35.54-90.292,35.54  C293.715,495.993,323.306,484.359,346.292,460.453"
style="fill:#BDBDBF;" />
  <path d="M288.832,429.286c-9.011,0-16.32-7.398-16.32-16.524s7.309-16.524,16.32-16.524
c8.999,0,16.308,7.398,16.308,16.524S297.831,429.286,288.832,429.286 M256.001,356.213v139.78l0,0
c37.715,0,67.304-11.635,90.292-35.54c1.082-1.124,2.148-2.276,3.2-3.455c13.925-15.6,24.135-34.676,32.456-54.715
C349.473,373.052,304.316,356.213,256.001,356.213" style="fill:#B19267;" />
  <path d="M308.018,266.551c-11.452,0-20.743-9.403-20.743-21.002c0-11.599,9.29-21.002,20.743-21.002
c11.438,0,20.728,9.403,20.728,21.002C328.745,257.148,319.457,266.551,308.018,266.551 M256.001,116.666v239.547
c48.315,0,93.473,16.838,125.947,46.07l0,0c4.58-11.029,8.594-22.346,12.32-33.544c4.053-12.177,7.965-22.589,11.748-32.659
c11.333-30.167,20.285-53.995,20.285-92.805c0-15.826-2.149-30.35-6.411-43.441c-2.104-6.462-4.722-12.573-7.851-18.32
c-1.201-2.205-2.478-4.359-3.828-6.455c-4.662-7.236-10.225-13.836-16.65-19.764c-8.143-7.514-17.671-13.952-28.514-19.251
c-9.809-4.795-20.155-8.419-30.603-11.153c-10.496-2.748-21.092-4.595-31.352-5.832C284.57,117.067,268.924,116.666,256.001,116.666  "
style="fill:#BD7F00;" />
</svg>
```

Animals

What animal does the object represent?

A: Elephant B: Giraffe
C: Lion D: Monkey

```
<svg id="Capa_1" style="enable-background:new 0 0 58 58;"
version="1.1" viewBox="0 0 58 58" x="0px" y="0px"
xml:space="preserve">
  <g>
   <rect height="58" style="fill:#CBB292;" width="44" x="1" />
   <rect height="58" style="fill:#7F6E5D;" width="8" x="1" />
   <rect height="12" style="fill:#EFEBDE;" width="22" x="16" y="10" />
   <rect height="2" style="fill:#D5D0BB;" width="14" x="20" y="13" />
   <rect height="2" style="fill:#D5D0BB;" width="14" x="20" y="17" />
   <g>
    <path
d="M45,41.2c0.969-2.183,3.109-4.2,5.684-4.2c3.467,0,5.964,2.821,6.278,
6.183
c0,0,0.17,0.835-0.203,2.337c-0.508,2.046-1.701,3.864-3.311,5.251L45,58l
-8.447-7.229c-1.61-1.387-2.803-3.205-3.311-5.251
c-0.373-1.502-0.203-2.337-0.203-2.337C33.352,39.821,35.849,37,39.316,
37C41.891,37,44.031,39.017,45,41.2z" style="fill:#F09372;" />
   </g>
  </g>
</svg>
```

Book

What type of object is depicted in the image?

A: notebook B: folder
C: book D: magazine

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <rect height="83.633" style="fill:#BCC987;" width="41.817" x="396.749" y="84.521" />
  <path d="M121.868,10.199c-23.858,0-43.199,19.341-43.199,43.199v405.203
c0,23.858,19.341,43.199,43.199,43.199h15.957V10.199H121.868z" style="fill:#FFAD61;" />
  <path d="M396.749,156.425c0-31.623,24.191-57.585,55.076-60.417v-42.61c0-23.858-19.341-43.199-43.199-43.199
H137.825v491.602h234.38c13.556,0,24.544-10.989,24.544-24.544V156.425z" style="fill:#FFE6B8;" />
  <g>
   <rect height="66.295" style="fill:#BCC987;" width="41.817" x="396.749" y="168.154" />
   <rect height="66.295" style="fill:#BCC987;" width="41.817" x="396.749" y="234.449" />
   <rect height="66.295" style="fill:#BCC987;" width="41.817" x="396.749" y="300.744" />
   <path d="M396.749,367.039h41.817v32.128c0,18.871-15.297,34.167-34.167,34.167h-7.649V367.039z" style="fill:#BCC987;" />
  </g>
  <path d="M312.854,270.441l-6.874,1.964l-7.07,4.718l-12.626,8.425
c-22.911-14.791-42.195-34.557-56.416-57.827l9.819-9.819l5.894-5.894l1.964-6.874l-46.158-46.158l-7.856,2.947
c-1.66,1.66-3.204,3.395-4.636,5.188c-19.383,24.265-17.839,59.741,4.636,82.217l75.128,75.128
c22.475,22.475,57.952,24.017,82.217,4.636c1.793-1.432,3.527-2.975,5.188-4.636l2.947-7.856L312.854,270.441z" style="fill:#D35B38;" />
  <g>
   <path d="M448.765,106.7c1.314-0.224,2.642-0.412,3.991-0.535c5.25-0.481,9.268-4.884,9.268-10.156v-42.61
C462.024,23.954,438.069,0,408.626,0H121.868C92.425,0,68.47,23.954,68.47,53.398v12.897h-8.295
c-5.633,0-10.199,4.567-10.199,10.199s4.566,10.199,10.199,10.199h8.295v338.614h-8.295c-5.633,0-10.199,4.567-10.199,10.199
c0,5.632,4.566,10.199,10.199,10.199h8.295v12.897c0,29.444,23.955,53.398,53.398,53.398h250.336
c19.157,0,34.744-15.585,34.744-34.744v-33.801c23.281-1.327,41.817-20.68,41.817-44.288V106.7z M88.869,458.602v-12.897h5.984
c5.633,0,10.199-4.567,10.199-10.199c0-5.632-4.566-10.199-10.199-10.199h-5.984V86.693h5.984c5.633,0,10.199-4.567,10.199-10.199
s-4.566-10.199-10.199-10.199h-5.984V53.398c0-18.195,14.803-33,33-33h5.757v471.203h-5.757
C103.672,491.602,88.869,476.797,88.869,458.602z M386.55,477.256c0,7.91-6.436,14.345-14.345,14.345H148.024V20.398h260.602
c18.195,0,33,14.804,33,33v33.927c-13.934,3.174-26.629,10.514-36.435,21.199c-12.021,13.1-18.641,30.111-18.641,47.902V477.256z
M428.367,290.545h-21.418v-45.896h21.418V290.545z M406.948,310.943h21.418v45.896h-21.418V310.943z M428.367,224.25h-21.418
v-45.896h21.418V224.25z M428.367,399.166c0,12.354-9.397,22.556-21.418,23.833v-45.762h21.418V399.166z M428.367,157.955h-21.418
v-1.53c0-17.11,8.366-32.241,21.418-41.373V157.955z" style="fill:#4D3D36;" />
   <path d="M357.246,337.057c2.118-1.693,4.147-3.506,6.03-5.388c1.03-1.03,1.828-2.269,2.339-3.633
l2.947-7.857c1.404-3.745,0.49-7.965-2.338-10.793l-46.158-46.158c-2.619-2.618-6.451-3.614-10.014-2.595l-6.874,1.964
c-1.016,0.29-1.981,0.737-2.86,1.324l-14.007,9.347c-16.955-11.92-31.779-26.997-43.413-44.152l9.896-9.897
c1.226-1.225,2.118-2.743,2.595-4.409l1.964-6.874c1.018-3.562,0.024-7.394-2.595-10.014l-46.158-46.158
c-2.828-2.826-7.047-3.741-10.794-2.338l-7.856,2.947c-1.364,0.511-2.602,1.309-3.632,2.339c-1.887,1.888-3.7,3.918-5.391,6.034
c-22.885,28.652-20.568,69.834,5.392,95.794l75.129,75.129c14.004,14.003,32.433,21.128,50.94,21.127
C328.191,352.793,344.049,347.599,357.246,337.057z M275.873,317.243l-75.128-75.129c-18.605-18.605-20.273-48.114-3.879-68.639
c0.68-0.851,1.387-1.682,2.114-2.486l36.598,36.598l-12.92,12.92c-3.331,3.332-3.947,8.511-1.49,12.53
c14.929,24.43,35.533,45.55,59.586,61.078c3.414,2.203,7.812,2.17,11.193-0.085l18.039-12.037l37.014,37.014
c-0.803,0.726-1.634,1.433-2.486,2.113C323.987,337.516,294.479,335.849,275.873,317.243z" style="fill:#4D3D36;" />
   <path d="M246.821,459.851h-5.1c-5.633,0-10.199,4.567-10.199,10.199c0,5.632,4.566,10.199,10.199,10.199h5.1
c5.633,0,10.199-4.567,10.199-10.199C257.02,464.419,252.454,459.851,246.821,459.851z" style="fill:#4D3D36;" />
   <path d="M364.112,459.851h-81.594c-5.633,0-10.199,4.567-10.199,10.199c0,5.632,4.566,10.199,10.199,10.199
h81.594c5.633,0,10.199-4.567,10.199-10.199C374.311,464.419,369.745,459.851,364.112,459.851z" style="fill:#4D3D36;" />
  </g>
</svg>
```

Book

What color is the telephone receiver on the object?

A: Red B: Blue
C: Green D: Yellow

```
<svg id="Layer_1" style="enable-background:new 0 0 504 504;" version="1.1" viewBox="0 0 504 504"
x="0px" y="0px" xml:space="preserve">
<circle cx="252" cy="252" r="252" style="fill:#84DBFF;" />
<path d="M188.5,137.6c0.5,1,0.9,2.1,1.4,3.1l0.1,0.1c3.2,7.2,6.4,14.3,9.7,21.4c4.4-1.9,8.9-3.7,13.4-5.1
c25.4-8.2,52.7-8.2,78,0c4.5,1.5,9,3.2,13.4,5.1c3.3-7.1,6.5-14.2,9.7-21.4c0-0.1,0.1-0.1,0.1-0.1c0.5-1,0.9-2.1,1.4-3
.1  C275.1,119.4,228.9,119.4,188.5,137.6z" style="fill:#ACB3BA;" />
<path d="M213,157c17.9,8.6,39,15,39,15s21.1-6.4,39-15C265.6,148.9,238.4,148.9,213,157z"
style="fill:#CED5E0;" />
<path d="M252,131.9c1.5,0,2.7-1.2,2.7-2.7v-8.1c0-14-6.3-27.6-17.4-37.5c-5.1-4.6-7.8-11.1-7.4-17.9
c0.7-11.1,9.6-20,20.7-20.7c6.2-0.4,12.1,1.7,16.6,6c4.5,4.2,7,10,7,16.1c0,1.5,1.2,2.7,2.7,2.7s2.7-1.2,2.7-2.7
c0-7.5-3.2-14.8-8.7-20c-5.6-5.2-12.9-7.9-20.5-7.4c-13.8,0.9-24.8,11.9-25.6,25.6c-0.5,8.5,2.8,16.6,9.1,22.2
c9.9,8.9,15.6,21.1,15.6,33.5v8.1C249.3,130.7,250.5,131.9,252,131.9z" style="fill:#324A5E;" />
<path d="M96.5,205.3c1.1,2.1,3.3,3.3,5.5,3.3h300c3.4,0,6.2-2.8,6.2-6.2c0-2.4-1.3-4.4-3.3-5.5l-150-81.7
c-1.9-1-4.2-1-6,0l-150,81.7C96,198.5,94.8,202.3,96.5,205.3z M252,142.8l120.5,53.3h-241L252,142.8z"
style="fill:#FFD05B;" />
<path
d="M27.8,246l66,68.7l38.4-30.2v189.3C167.9,493.1,208.6,504,252,504c43.3,0,84.1-10.9,119.7-30.2V284.5
l38.4,30.2l66-68.7l-77-73.2c-6.7-6.4-15.3-10.7-24.5-12.4l-61.8-11c0.2,3.3-1.8,6.6-6,6.6c-0.6,0-1.1,0-1.5-0.2
c-5.9,19-27.6,33.1-53.4,33.3c-26.1-0.2-47.9-14.6-53.6-33.9c-0.8,0.5-1.9,0.8-3.1,0.8c-4.1,0-6.1-3.2-6.1-6.3l-60,1
0.7  c-9.2,1.7-17.7,5.9-24.5,12.4L27.8,246z" style="fill:#FF7058;" />
<polygon points="390.9,299.5 410.2,314.7 476.2,246 458.5,229.2 " style="fill:#324A5E;" />
<g>
      <polygon points="390.9,299.5 410.2,314.7 476.2,246 458.5,229.2  " style="fill:#E6E9EE;" />
      <polygon points="27.8,246 93.8,314.7 113.1,299.5 45.5,229.2  " style="fill:#E6E9EE;" />
</g>
<rect height="107.3" style="fill:#F1543F;" width="32.1" x="235.9" y="172.1" />
<g>
      <path d="M195.4,189.7l26.9,29l29.6-46.6c0,0-21.1-6.4-39-15c-10.2-5-19.5-10.6-23-16.3l-0.1-0.1
c-0.7-1-1.1-2.1-1.4-3.1l-3.5,8.6C179.3,160.2,182.5,175.7,195.4,189.7z" style="fill:#E6E9EE;" />
      <path d="M252,172.1l29.6,46.6l27-29c13-14,16.2-29.4,10.4-43.5l-3.5-8.6c-0.2,1-0.7,2.1-1.4,3.1
c0,0.1-0.1,0.1-0.1,0.1c-3.6,5.6-12.9,11.3-23.1,16.2C273.1,165.7,252,172.1,252,172.1z" style="fill:#E6E9EE;" />
      <circle cx="252" cy="232.3" r="8.3" style="fill:#E6E9EE;" />
      <circle cx="252" cy="263.6" r="8.3" style="fill:#E6E9EE;" />
</g></svg>
```

Clothing

What is the color of the collar of the shirt in the image?

A: Red B: Yellow
C: Blue D: Gray

```
<svg id="Layer_1" style="enable-background:new 0 0 503.739 503.739;" version="1.1" viewBox="0 0 503.739 503.739" x="0px" y="0px"
xml:space="preserve">
  <path d="M16.684,167.83c-9.233,20.984-15.948,85.613,8.393,117.508v33.574c0,0,19.305,16.787,50.361,16.787
c50.361,0,124.223-8.393,159.475-8.393s117.508,22.662,194.728-6.715c0,0,42.807-10.072,56.236-16.787
c12.59-6.715,23.502-47.003-37.771-63.79c-62.111-16.787-104.079-38.61-104.079-38.61l-16.787,33.574
C327.241,234.978,66.205,195.529,16.684,167.83" style="fill:#FFE100;" />
  <path d="M25.077,318.912c0,0,19.305,16.787,50.361,16.787c50.361,0,124.223-8.393,159.475-8.393
s117.508,22.662,194.728-6.715c0,0,42.807-10.072,56.236-16.787c7.554-4.197,14.269-20.144,5.036-36.092
c-3.357,1.679-5.036,2.518-5.036,2.518c-12.59,6.715-56.236,16.787-56.236,16.787c-78.059,29.377-159.475,6.715-194.728,6.715
s-109.115,8.393-159.475,8.393c-31.056,0-50.361-16.787-50.361-16.787V318.912z" style="fill:#FDCC00;" />
  <g>
   <path d="M50.258,318.912v-33.574c-20.984-28.538-18.466-81.416-10.911-108.275
c-10.072-3.357-17.626-6.715-22.662-9.233c-9.233,20.984-15.948,85.613,8.393,117.508v33.574c0,0,19.305,16.787,50.361,16.787
c5.875,0,12.59,0,19.305,0C67.045,333.181,50.258,318.912,50.258,318.912" style="fill:#FFFFFF;" />
   <path d="M50.258,318.912v-20.984c-15.948-5.036-25.18-12.59-25.18-12.59v33.574
c0,0,19.305,16.787,50.361,16.787c5.875,0,12.59,0,19.305,0C67.045,333.181,50.258,318.912,50.258,318.912" style="fill:#FFFFFF;" />
  </g>
  <g>
   <path d="M447.267,240.014c-61.272-16.787-103.239-38.61-103.239-38.61l-5.036,9.233
c18.466,7.554,47.003,19.305,83.095,29.377c61.272,16.787,51.2,57.075,37.77,63.79c-12.59,6.715-56.236,16.787-56.236,16.787
c-26.02,10.072-52.879,13.43-78.059,14.269c31.895,0.839,67.987-1.679,103.239-14.269c0,0,42.807-10.072,56.236-16.787
S508.54,256.801,447.267,240.014" style="fill:#FFA800;" />
   <path d="M490.913,267.712c-3.357,1.679-5.036,2.518-5.036,2.518c-3.357,1.679-9.233,4.197-15.948,6.715
c2.518,12.59-2.518,23.502-9.233,27.698c-12.59,6.715-56.236,16.787-56.236,16.787c-26.02,10.072-52.879,13.43-78.059,14.269
c31.895,0.839,67.987-1.679,103.239-14.269c0,0,42.807-10.072,56.236-16.787C493.431,299.607,500.146,283.66,490.913,267.712"
style="fill:#FFA800;" />
  </g>
  <path d="M339.831,344.093c-26.859,0-52.039-3.357-71.344-5.036c-13.43-1.679-25.18-3.357-33.574-3.357h-0.839
c-15.948,0-40.289,1.679-66.308,3.357c-30.216,2.518-64.63,5.036-92.328,5.036c-34.413,0-55.397-18.466-56.236-19.305
c-1.679-0.839-2.518-3.357-2.518-5.875v-31.056c-25.18-36.092-16.787-102.4-7.554-123.384c0.839-2.518,2.518-4.197,5.036-4.197
c2.518-0.839,5.036-0.839,6.715,0c43.646,24.341,262.715,58.754,301.325,65.469l14.269-27.698c0.839-1.679,2.518-3.357,5.036-4.197
s5.036-0.839,6.715,0c0,0,41.967,21.823,101.561,37.77c43.646,11.751,52.879,34.413,53.718,47.003
c1.679,15.108-5.875,27.698-14.269,32.734c-13.43,6.715-51.2,15.948-57.915,17.626C404.461,339.056,374.245,344.093,339.831,344.093  z
M234.074,318.912h0.839c9.233,0,20.984,1.679,34.413,3.357c19.305,2.518,43.646,5.036,69.666,5.036
c32.734,0,61.272-5.036,87.292-14.269h0.839c11.751-2.518,44.485-10.911,53.718-15.948c2.518-0.839,6.715-7.554,5.036-15.948
c-0.839-8.393-7.554-23.502-41.967-32.734c-47.003-12.59-82.256-28.538-97.364-35.252l-13.43,26.02
c-1.679,3.357-5.036,5.036-8.393,4.197c-8.393-1.679-237.534-36.092-303.003-63.79c-6.715,24.341-9.233,75.541,10.072,100.721
c0.839,1.679,1.679,3.357,1.679,5.036v29.377c5.875,4.197,20.984,12.59,41.967,12.59c26.859,0,61.272-2.518,90.649-5.036
C192.946,320.591,217.287,318.912,234.074,318.912z" />
  <path d="M339.831,344.093c-26.859,0-52.039-3.357-71.344-5.036c-13.43-1.679-25.18-3.357-33.574-3.357h-0.839
c-15.948,0-40.289,1.679-66.308,3.357c-30.216,2.518-64.63,5.036-92.328,5.036c-34.413,0-55.397-18.466-56.236-19.305
c-1.679-0.839-2.518-3.357-2.518-5.875v-33.574c0-3.357,1.679-6.715,5.036-7.554c3.357-1.679,6.715-0.839,9.233,0.839
c0,0,17.626,15.108,44.485,15.108s61.272-2.518,90.649-5.036c26.02-1.679,50.361-3.357,67.148-3.357h1.679
c9.233,0,20.984,1.679,34.413,3.357c19.305,2.518,43.646,5.036,69.666,5.036c32.734,0,61.272-5.036,87.292-14.269h0.839
c11.751-2.518,44.485-10.911,53.718-15.948c0,0,2.518-0.839,5.875-2.518c4.197-1.679,8.393-0.839,10.911,3.357
c5.875,9.233,6.715,20.144,4.197,30.216c-2.518,7.554-6.715,14.269-12.59,17.626c-13.429,6.715-51.2,15.948-57.915,17.626
C404.461,339.056,374.245,344.093,339.831,344.093z M234.074,318.912h0.839c9.233,0,20.984,1.679,34.413,3.357
c19.305,2.518,43.646,5.036,69.666,5.036c32.734,0,61.272-5.036,87.292-14.269h0.839c11.751-2.518,44.485-10.911,53.718-15.948
c0.839-0.839,3.357-3.357,5.036-7.554c0.839-2.518,0.839-5.875,0-10.072c-15.108,6.715-48.682,15.108-54.557,15.948
c-27.698,10.072-57.915,15.108-92.328,15.108c-26.859,0-52.039-3.357-71.344-5.036c-12.59-1.679-24.341-3.357-32.734-3.357h-0.839
c-15.948,0-40.289,1.679-66.308,3.357c-30.216,2.518-64.63,5.036-92.328,5.036c-17.626,0-32.734-5.036-41.967-10.072v14.269
c5.875,4.197,20.984,12.59,41.967,12.59c26.859,0,61.272-2.518,90.649-5.036C192.946,320.591,217.287,318.912,234.074,318.912z" />
</svg>
```

Clothing

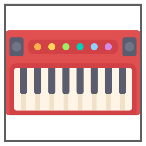
What type of object is depicted in the image?

A: Hat B: Shoe
C: Bag D: Glove

Figure 18: SVG examples in our SGP-Bench.

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px" xml:space="preserve">
<path d="M494.345,414.897H17.655C7.904,414.897,0,406.993,0,397.241V114.759  c0-9.751,7.904-17.655,17.655-17.655h476.69c9.751,0,17.655,7.904,17.655,17.655v282.483
C512,406.993,504.096,414.897,494.345,414.897z" style="fill:#E15050;" />
 <g>
  <path d="M406.069,185.379H105.931c-9.751,0-17.655-7.904-17.655-17.655v-17.655   c0-9.751,7.904-17.655,17.655-17.655h300.138c9.751,0,17.655,7.904,17.655,17.655v17.655
C423.724,177.475,415.82,185.379,406.069,185.379z" style="fill:#D24146;" />
  <path d="M494.345,414.897H17.655V247.172c0-14.626,11.857-26.483,26.483-26.483h423.724   c14.626,0,26.483,11.857,26.483,26.483V414.897z" style="fill:#D24146;" />
 </g>
 <path d="M467.862,397.241H44.138c-4.875,0-8.828-3.953-8.828-8.828V247.172c0-4.875,3.953-8.828,8.828-8.828
h423.724c4.875,0,8.828,3.953,8.828,8.828v141.241C476.69,393.289,472.737,397.241,467.862,397.241z" style="fill:#FFF5E6;" />
 <circle cx="229.517" cy="158.897" r="13.241" style="fill:#A0EB64;" />
 <circle cx="176.552" cy="158.897" r="13.241" style="fill:#FFD558;" />
 <circle cx="123.586" cy="158.897" r="13.241" style="fill:#FFAF4B;" />
 <circle cx="282.483" cy="158.897" r="13.241" style="fill:#00D2B9;" />
 <circle cx="335.448" cy="158.897" r="13.241" style="fill:#8CD2FF;" />
 <circle cx="388.414" cy="158.897" r="13.241" style="fill:#D28CE6;" />
 <g>
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="114.759" y="326.621" />
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="167.724" y="326.621" />
 </g>
 <g>
  <path d="M128,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C136.828,331.496,132.875,335.448,128,335.448z" style="fill:#5B5D6E;" />
  <path d="M180.966,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C189.793,331.496,185.841,335.448,180.966,335.448z" style="fill:#5B5D6E;" />
 </g>
 <g>
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="220.69" y="326.621" />
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="273.655" y="326.621" />
 </g>
 <g>
  <path d="M233.931,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C242.759,331.496,238.806,335.448,233.931,335.448z" style="fill:#5B5D6E;" />
  <path d="M286.897,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C295.724,331.496,291.772,335.448,286.897,335.448z" style="fill:#5B5D6E;" />
 </g>
 <g>
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="326.621" y="326.621" />
  <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="379.586" y="326.621" />
 </g>
 <g>
  <path d="M339.862,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C348.69,331.496,344.737,335.448,339.862,335.448z" style="fill:#5B5D6E;" />
  <path d="M392.828,335.448H384c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C401.655,331.496,397.703,335.448,392.828,335.448z" style="fill:#5B5D6E;" />
 </g>
 <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="432.552" y="326.621" />
 <path d="M445.793,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276  C454.621,331.496,450.668,335.448,445.793,335.448z" style="fill:#5B5D6E;" />
 <rect height="70.621" style="fill:#F0E1C8;" width="17.655" x="61.793" y="326.621" />
 <g>
  <path d="M75.034,335.448h-8.828c-4.875,0-8.828-3.953-8.828-8.828v-88.276h26.483v88.276   C83.862,331.496,79.91,335.448,75.034,335.448z" style="fill:#5B5D6E;" />
  <path d="M61.793,194.207h-35.31c-4.875,0-8.828-3.953-8.828-8.828v-52.966c0-4.875,3.953-8.828,8.828-8.828
h35.31c4.875,0,8.828,3.953,8.828,8.828v52.966C70.621,190.254,66.668,194.207,61.793,194.207z" style="fill:#5B5D6E;" />
  <path d="M485.517,194.207h-35.31c-4.875,0-8.828-3.953-8.828-8.828v-52.966c0-4.875,3.953-8.828,8.828-8.828
h35.31c4.875,0,8.828,3.953,8.828,8.828v52.966C494.345,190.254,490.392,194.207,485.517,194.207z" style="fill:#5B5D6E;" />
 </g>
 <g>
  <circle cx="44.138" cy="158.897" r="17.655" style="fill:#707487;" />
  <circle cx="467.862" cy="158.897" r="17.655" style="fill:#707487;" />
 </g>
</svg>
```

Musical Instrument



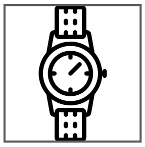
How many black keys are visible on the object?

A: 5 B: 6

C: 7 D: 8

```
<svg id="Layer_1" style="enable-background:new 0 0 511.999 511.999;" version="1.1" viewBox="0 0 511.999 511.999" x="0px" y="0px"
xml:space="preserve">
 <g><g>
   <path d="M247.448,171.023c-46.856,0-84.977,38.121-84.977,84.977s38.121,84.977,84.977,84.977s84.977-38.121,84.977-84.977
S294.304,171.023,247.448,171.023z M255.465,324.463v-8.605c0-4.427-3.589-8.017-8.017-8.017s-8.017,3.589-8.017,8.017v8.605
c-31.618-3.68-56.766-28.827-60.446-60.446h8.605c4.427,0,8.017-3.589,8.017-8.017s-3.589-8.017-8.017-8.017h-8.605
c3.68-31.618,28.827-56.766,60.446-60.446v8.605c0,4.427,3.589,8.017,8.017,8.017s8.017-3.589,8.017-8.017v-8.605
c31.618,3.68,56.766,28.827,60.446,60.446h-8.605c-4.427,0-8.017,3.589-8.017,8.017s3.589,8.017,8.017,8.017h8.605
C312.231,295.635,287.083,320.782,255.465,324.463z" />
  </g> </g><g><g>
   <path d="M287.321,216.127c-3.13-3.131-8.207-3.131-11.337,0l-34.205,34.205c-3.131,3.131-3.131,8.207,0,11.337
c1.565,1.565,3.617,2.348,5.668,2.348s4.103-0.782,5.668-2.348l34.205-34.205C290.452,224.333,290.452,219.257,287.321,216.127z" />
  </g></g><g><g>
   <path d="M230.348,51.307c-4.427,0-8.017,3.589-8.017,8.017v8.552c0,4.427,3.589,8.017,8.017,8.017s8.017-3.589,8.017-8.017v-8.552
C238.365,54.896,234.776,51.307,230.348,51.307z" />
  </g></g><g><g>
   <path d="M264.553,51.307c-4.427,0-8.017,3.589-8.017,8.017v8.552c0,4.427,3.589,8.017,8.017,8.017s8.017-3.589,8.017-8.017v-8.552
C272.569,54.896,268.98,51.307,264.553,51.307z" />
  </g></g><g><g>
   <path d="M230.348,436.109c-4.427,0-8.017,3.589-8.017,8.017v8.552c0,4.427,3.589,8.017,8.017,8.017s8.017-3.589,8.017-8.017
v-8.552C238.365,439.698,234.776,436.109,230.348,436.109z" />
  </g></g><g><g>
   <path d="M264.553,436.109c-4.427,0-8.017,3.589-8.017,8.017v8.552c0,4.427,3.589,8.017,8.017,8.017s8.017-3.589,8.017-8.017
v-8.552C272.569,439.698,268.98,436.109,264.553,436.109z" />
  </g></g><g><g>
   <path d="M375.716,239.432c-4.427,0-8.017,3.589-8.017,8.017v0.534h-1.345c-1.763-26.406-12.16-50.484-28.392-69.427l-12.92-70.968
c-1.464-7.838-8.314-13.526-16.287-13.526h-1.984V16.568C306.772,7.432,299.34,0,290.204,0h-85.511
c-9.136,0-16.568,7.432-16.568,16.568v77.495h-1.984c-7.972,0-14.821,5.688-16.293,13.562l-12.913,70.932
c-17.861,20.844-28.668,47.905-28.668,77.443s10.807,56.599,28.668,77.443l12.92,70.968c1.464,7.837,8.314,13.525,16.286,13.525
h1.985v77.495c0,9.136,7.432,16.568,16.568,16.568h85.511c9.136,0,16.568-7.432,16.568-16.568v-77.494h1.984
c7.972,0,14.821-5.688,16.293-13.562l12.913-70.932c16.232-18.944,26.629-43.022,28.392-69.427h1.345v0.534
c0,4.427,3.589,8.017,8.017,8.017c4.427,0,8.017-3.589,8.017-8.017v-17.102C383.732,243.021,380.143,239.432,375.716,239.432z
M204.158,16.568c0-0.295,0.239-0.534,0.534-0.534h17.666c-0.012,0.177-0.027,0.354-0.027,0.534v8.552
c0,4.427,3.589,8.017,8.017,8.017c4.427,0,8.017-3.589,8.017-8.017v-8.552c0-0.181-0.015-0.357-0.027-0.534h18.225
c-0.012,0.177-0.027,0.354-0.027,0.534v8.552c0,4.427,3.589,8.017,8.017,8.017c4.427,0,8.017-3.589,8.017-8.017v-8.552
c0-0.181-0.015-0.357-0.027-0.534h17.661c0.295,0,0.534,0.239,0.534,0.534v94.063c0,0.295-0.239,0.534-0.534,0.534h-17.661
c0.012-0.177,0.027-0.354,0.027-0.534v-8.552c0-4.427-3.589-8.017-8.017-8.017s-8.017,3.589-8.017,8.017v8.552
c0,0.181,0.015,0.357,0.027,0.534h-18.225c0.012-0.177,0.027-0.354,0.027-0.534v-8.552c0-4.427-3.589-8.017-8.017-8.017
s-8.017,3.589-8.017,8.017v8.552c0,0.181,0.015,0.357,0.027,0.534h-17.666c-0.295,0-0.534-0.239-0.534-0.534V16.568z
M185.616,110.533c0.046-0.253,0.267-0.437,0.525-0.437h1.984v0.534c0,9.136,7.432,16.568,16.568,16.568h85.511
c9.136,0,16.568-7.432,16.568-16.568v-0.534h1.984c0.258,0,0.478,0.184,0.518,0.4l9.058,49.754
c-19.825-14.714-44.354-23.431-70.884-23.431s-51.059,8.717-70.884,23.431L185.616,110.533z M290.739,495.432
c0,0.295-0.239,0.534-0.534,0.534h-17.661c0.012-0.177,0.027-0.354,0.027-0.534v-8.552c0-4.427-3.589-8.017-8.017-8.017
s-8.017,3.589-8.017,8.017v8.552c0,0.181,0.015,0.357,0.027,0.534h-18.225c0.012-0.177,0.027-0.354,0.027-0.534v-8.552
c0-4.427-3.589-8.017-8.017-8.017s-8.017,3.589-8.017,8.017v8.552c0,0.181,0.015,0.357,0.027,0.534h-17.666
c-0.295,0-0.534-0.239-0.534-0.534V401.37c0-0.295,0.239-0.534,0.534-0.534h17.666c-0.012,0.177-0.027,0.354-0.027,0.534v8.552
c0,4.427,3.589,8.017,8.017,8.017c4.427,0,8.017-3.589,8.017-8.017v-8.552c0-0.181-0.015-0.357-0.027-0.534h18.225
c-0.012,0.177-0.027,0.354-0.027,0.534v8.552c0,4.427,3.589,8.017,8.017,8.017c4.427,0,8.017-3.589,8.017-8.017v-8.552
c0-0.181-0.015-0.357-0.027-0.534h17.661c0.295,0,0.534,0.239,0.534,0.534V495.432z M309.281,401.467
c-0.046,0.253-0.267,0.437-0.525,0.437h-1.984v-0.534c0-9.136-7.432-16.568-16.568-16.568h-85.511
c-9.136,0-16.568,7.432-16.568,16.568v0.534h-1.984c-0.258,0-0.478-0.184-0.518-0.4l-9.058-49.754
c19.825,14.714,44.354,23.431,70.884,23.431s51.059-8.717,70.884-23.431L309.281,401.467z M247.448,359.148
C190.573,359.148,144.3,312.876,144.3,256s46.272-103.148,103.148-103.148S350.597,199.124,350.597,256    S304.324,359.148,247.448,359.148z" />
  </g>
 </g>
</svg>
```

Time Clock



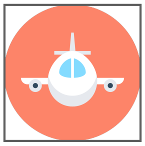
How many buttons are visible on the object?

A: 1 B: 2

C: 3 D: 4

```
<svg id="Layer_1" style="enable-background:new 0 0 504 504;" version="1.1"
viewBox="0 0 504 504" x="0px" y="0px" xml:space="preserve">
  <circle cx="252" cy="252" r="252" style="fill:#FD8469;" />
  <polygon points="262.8,171.6 256.7,103.7 247.3,103.7 241.2,171.6 184.9,171.6
184.9,185.3 319.1,185.3   319.1,171.6 " style="fill:#E6E9EE;" />
  <g>
    <path
d="M417.8,294.7H86.2c-13.2,0-23.9-10.7-23.9-23.9l0,0h379.4l0,0C441.7,284,431,294.7,4
17.8,294.7z" style="fill:#FFFFFF;" />
    <path
d="M342.8,280.4c0,56.6-40.7,96.2-90.8,96.2s-90.8-39.6-90.8-96.2S212,171.7,252,171.7
S342.8,223.8,342.8,280.4z" style="fill:#FFFFFF;" />
  </g>
  <path
d="M300.8,250.6c0,27-97.7,27-97.7,0s27.6-52.3,48.8-52.3C271.3,198.3,300.8,223.6,300.8
,250.6z" style="fill:#84DBFF;" />
  <rect height="88.4" style="fill:#FFFFFF;" width="10.3" x="246.9" y="194.9" />
  <g>
    <path
d="M252,337c-43.8,0-80.4-30.2-88.9-75.6c-1.2,6.2-1.9,12.6-1.9,19c0,56.6,40.7,96.2,90.8,9
6.2   s90.8-39.6,90.8-96.2c0-6.4-0.7-12.8-1.9-19C332.4,306.8,295.8,337,252,337z"
style="fill:#E6E9EE;" />
    <circle cx="391.1" cy="299.4" r="23.7" style="fill:#E6E9EE;" />
    <circle cx="115.5" cy="299.4" r="23.7" style="fill:#E6E9EE;" />
  </g>
  <g>
    <circle cx="391.1" cy="299.4" r="9.5" style="fill:#324A5E;" />
    <circle cx="115.5" cy="299.4" r="9.5" style="fill:#324A5E;" />
  </g>
</svg>
```

Aerial Crafts



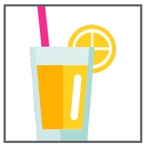
What is the object in the image?

A: Airplane B: Boat

C: Car D: Train

```
<svg id="Layer_1" style="enable-background:new 0 0 155.883 155.883;" version="1.1" viewBox="0 0 155.883
155.883" x="0px" y="0px" xml:space="preserve">
 <g><g><g>
 <circle cx="99.422" cy="50.126" r="27.492" style="fill:#FFD01F;" />
 </g><g>
 <circle cx="99.804" cy="49.744" r="22.528" style="fill:#FFD01F;" />
 </g>
 <g>
  <circle cx="99.804" cy="49.744" r="22.528" style="fill:#FFFFFF;" />
 </g>
 <g><g>
   <path d="M98.754,48.472V31.184c-9.616,0.434-17.406,7.843-18.451,17.288H98.754z" style="fill:#FFD01F;" />
 </g><g>
 <path d="M119.305,48.217c-1.584-9.02-8.962-16.036-18.166-17.056v17.056H119.305z" style="fill:#FFD01F;" />
 </g><g>
  <path d="M98.754,51.018v17.287c-9.616-0.434-17.406-7.842-18.451-17.287
C80.303,51.018,98.754,51.018,98.754,51.018z" style="fill:#FFD01F;" />
 </g><g>
  <path d="M119.305,51.271c-1.584,9.02-8.962,16.037-18.166,17.056V51.271H119.305z" style="fill:#FFD01F;" />
 </g></g></g><g>
  <polygon points="94.188,155.883 37.438,155.883 28.969,47.883 102.656,47.883    " style="fill:#ABD9D5;" />
 </g><g>
  <polygon points="63.816,155.883 37.438,155.883 28.969,47.883 63.816,47.883    " style="fill:#CDE8E6;" />
 </g><g>
  <polygon points="88.258,141.883 43.367,141.883 34.898,47.883 96.729,47.883    " style="fill:#DDF0EE;" />
 </g><g>
  <polygon points="36.805,68.606 43.367,141.449 88.258,141.449 94.82,68.606    " style="fill:#FFD01F;" />
 </g>
 <path d="M83.676,127.346c-0.216,2.329-2.278,4.044-4.607,3.828l0,0c-2.328-0.215-4.042-2.277-3.827-4.606
l4.048-43.857c0.215-2.329,2.277-4.043,4.605-3.828l0,0c2.33,0.215,4.043,2.277,3.828,4.606L83.676,127.346z"
style="fill:#FFFFFF;" />
 <g><g>
  <polygon points="66.236,68.606 36.805,68.606 43.367,141.449 66.236,141.449    " style="fill:#FFB000;" />
 </g><g>
  <polygon points="66.236,68.606 36.805,68.606 43.367,141.449 66.236,141.449    " style="fill:#FFB000;" />
 </g></g><g><g>
  <polygon points="37.193,0 42.182,47.433 51.064,47.433 46.076,0    " style="fill:#F62D8D;" />
 </g></g></g>
</svg>
```

Beverage



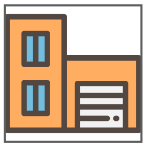
What is the primary color of the liquid in the glass?

A: Yellow B: Blue

C: Red D: Green

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512"
x="0px" y="0px" xml:space="preserve">
 <g>
  <rect height="424.47" style="fill:#FFAD61;" width="212.755" x="10.199" y="43.765" />
  <rect height="270.585" style="fill:#FFAD61;" width="278.846" x="222.955" y="197.65" />
 </g>
 <rect height="174.539" style="fill:#F2F2F2;" width="179.873" x="272.441" y="293.696" />
 <g>
  <rect height="114.639" style="fill:#72BEDE;" width="86.754" x="73.2" y="290.596" />
  <rect height="114.639" style="fill:#72BEDE;" width="86.754" x="73.2" y="106.765" />
 </g>
 <g>
  <path
d="M501.801,478.435c5.632,0,10.199-4.567,10.199-10.199V197.648c0-5.632-4.567-10.199-10.199-10.199
h-268.65V43.764c0-5.632-4.567-10.199-10.199-10.199H10.199C4.567,33.565,0,38.132,0,43.764v424.473
c0,5.632,4.567,10.199,10.199,10.199H501.801z M212.752,458.037H20.398V53.963h192.354V458.037z
M282.64,348.305h159.471v22.462   H282.64V348.305z
M442.112,327.906H282.64v-24.011h159.471v24.011H442.112z M282.64,458.037v-24.012h101.424
c5.632,0,10.199-4.567,10.199-10.199c0-5.632-4.567-10.199-10.199-10.199H282.64v-22.461h159.471v66.871
H282.64z M491.602,458.037
H462.51V293.696c0-5.632-4.567-10.199-10.199-10.199h-179.87c-5.632,0-10.199,4.567-10.199,10.199v164.3
41h-29.091v-250.19h258.451   V458.037z" style="fill:#534741;" />
  <path d="M159.952,280.399H73.198c-5.632,0-10.199,4.567-10.199,10.199v114.638
c0,5.632,4.567,10.199,10.199,10.199h86.754c5.632,0,10.199-4.567,10.199-10.199V290.598
C170.151,284.965,165.585,280.399,159.952,280.399z M83.398,300.797h22.978v94.24H83.398V300.797z
M149.753,395.037h-22.979v-94.24   h22.979V395.037z" style="fill:#534741;" />
  <path d="M159.952,96.565H73.198c-5.632,0-10.199,4.567-10.199,10.199v114.638
c0,5.632,4.567,10.199,10.199,10.199h86.754c5.632,0,10.199-4.567,10.199-10.199V106.764
C170.151,101.131,165.585,96.565,159.952,96.565z M83.398,116.963h22.978v94.24H83.398V116.963z
M149.753,211.203h-22.979v-94.24   h22.979V211.203z" style="fill:#534741;" />
  <path
d="M419.178,413.627h-3.097c-5.632,0-10.199,4.567-10.199,10.199c0,5.632,4.567,10.199,10.199,10.199
h3.097c5.632,0,10.199-4.567,10.199-10.199C429.377,418.194,424.811,413.627,419.178,413.627z"
style="fill:#534741;" />
 </g>
</svg>
```

Building



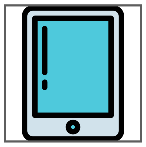
What is the shape of the windows on the left side of the building?

A: Circular B: Triangular

C: Rectangular D: Hexagonal

```
<svg id="Layer_1" style="enable-background:new 0 0 502 502;" version="1.1" viewBox="0 0 502 502"
x="0px" y="0px" xml:space="preserve">
  <g>
    <g>
      <path d="M72.898,461.62c0,16.779,13.602,30.38,30.38,30.38h295.444c16.779,0,30.38-13.602,30.38-30.38
V40.38c0-16.778-13.602-30.38-30.38-30.38H103.278c-16.779,0-30.38,13.602-30.38,30.38V461.62z"
style="fill:#D1E2EB;" />
      <path
d="M398.722,502H103.278c-22.266,0-40.38-18.114-40.38-40.38V40.38C62.898,18.114,81.013,0,103.278,0h29
5.443    c22.266,0,40.38,18.114,40.38,40.38v421.24C439.102,483.886,420.987,502,398.722,502z M103.278,20
c-11.238,0-20.38,9.143-20.38,20.38v421.24c0,11.237,9.143,20.38,20.38,20.38h295.443c11.238,0,20.38-9.143,2
0.38-20.38V40.38    c0-11.237-9.143-20.38-20.38-20.38C398.721,20,103.278,20,103.278,20z" />
    </g>
    <g>
     <rect height="356.565" style="fill:#4EC9DC;" width="280.204" x="110.898" y="48" />
     <path
d="M391.102,414.565H110.898c-5.523,0-10-4.477-10-10V48c0-5.523,4.477-10,10-10h280.204c5.523,0,10,4.4
77,10,10v356.565    C401.102,410.088,396.625,414.565,391.102,414.565z
M120.898,394.565h260.204V58H120.898V394.565z" />
    </g>
    <g>
      <path
d="M147,257c-5.523,0-10-4.477-10-10V86c0-5.523,4.477-10,10-10s10,4.477,10,10v161C157,252.523,152.523
,257,147,257z" />
    </g>
    <g>
      <path
d="M147,312c-5.523,0-10-4.477-10-10v-19c0-5.523,4.477-10,10-10s10,4.477,10,10v19C157,307.523,152.523,
312,147,312z" />
    </g>
    <g>
      <circle cx="251" cy="448" r="17" style="fill:#4EC9DC;" />
      <path d="M251,475c-14.888,0-27-12.112-27-27s12.112-27,27-27s27,12.112,27,27S265.888,475,251,475z
M251,441c-3.86,0-7,3.14-7,7    s3.14,7,7,7s7-3.14,7-7S254.86,441,251,441z" />
    </g>
  </g>
</svg>
```

Computer



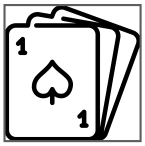
What type of object is shown in the image?

A: Tablet B: Laptop

C: Smartphone D: Television

```
<svg id="Layer_1" style="enable-background:new 0 0 512.016 512.016;" version="1.1" viewBox="0 0 512.016 512.016"
x="0px" y="0px" xml:space="preserve">
 <g>
  <g>
   <g>
    <path
d="M307.215,443.749v-42.667c0-3.448-2.074-6.562-5.265-7.885c-3.191-1.314-6.852-0.589-9.301,1.852l-17.067,17.067
c-3.337,3.337-3.337,8.73,0,12.066c3.328,3.337,8.73,3.337,12.066,0l2.5-2.5v22.067c-4.719,0-8.533,3.823-8.533,8.533
c0,4.71,3.814,8.533,8.533,8.533h17.067c4.71,0,8.533-3.823,8.533-8.533C315.749,447.572,311.926,443.749,307.215,443.749
z" />
    <path
d="M76.815,170.683v-42.667c0-3.447-2.074-6.562-5.274-7.885c-3.191-1.323-6.852-0.589-9.293,1.852L45.182,139.05
c-3.337,3.337-3.337,8.73,0,12.066c3.328,3.337,8.73,3.337,12.066,0l2.5-2.5v22.067c-4.719,0-8.533,3.823-8.533,8.533
s3.814,8.533,8.533,8.533h17.067c4.71,0,8.533-3.823,8.533-8.533S81.526,170.683,76.815,170.683z" />
    <path
d="M484.086,95.316L229.332,2.576c-10.65-3.874-22.187-3.362-32.444,1.425c-10.274,4.787-18.065,13.286-21.82,23.62
l-2.62,6.426l-41.421-7.1c-11.23-1.971-22.545,0.538-31.881,7.074c-9.344,6.536-15.573,16.324-17.536,27.486l-1.237,6.775H4
2.682     c-23.526,0-42.667,19.14-42.667,42.667v358.4c0,23.526,19.14,42.667,42.667,42.667h273.067h8.533h8.533
c23.236,0,45.969-3.388,54.921-27.955l121.711-334.379C517.434,127.709,506.059,103.321,484.086,95.316z
M341.349,469.349
c0,14.114-11.486,25.6-25.6,25.6H42.682c-14.123,0-25.6-11.486-25.6-25.6v-358.4c0-14.114,11.477-25.6,25.6-25.6h44.8
c4.113,0,7.646-2.944,8.388-6.997l2.543-13.824c1.186-6.733,4.915-12.604,10.513-16.529c5.623-3.917,12.416-5.436,19.174-4
.233     l48.205,8.26c3.891,0.631,7.817-1.476,9.344-5.18l5.342-13.073c2.313-6.366,6.963-11.443,13.107-14.302
c6.127-2.867,13.005-3.174,19.396-0.853l131.277,47.787L222.531,43.084c-4.651-0.819-9.071,2.278-9.882,6.921
c-0.828,4.642,2.27,9.071,6.912,9.89l47.573,8.388H119.473c-4.719,0-8.533,3.823-8.533,8.533c0,4.71,3.814,8.533,8.533,8.53
3     h196.275c14.114,0,25.6,11.486,25.6,25.6V469.349z
M347.228,82.405l49.749,8.772c13.901,2.449,23.219,15.753,20.779,29.653
l-59.341,336.555V110.949C358.415,99.941,354.106,89.983,347.228,82.405z
M493.413,143.854l-121.02,332.467l62.165-352.529
c1.954-11.102-0.691-21.922-6.417-30.686l50.116,18.244C491.382,116.129,498.183,130.713,493.413,143.854z" />
    <path d="M222.07,245.853c-12.826-10.394-26.086-21.146-35.541-36.898c-3.081-5.137-11.546-5.137-14.635,0
c-9.728,16.222-22.067,27.145-34.005,37.7c-18.244,16.145-35.473,31.394-35.473,60.561c0,23.526,19.14,42.667,42.667,42.66
7     c11.443,0,19.439-3.004,25.6-7.842v24.909c0,4.71,3.814,8.533,8.533,8.533c4.71,0,8.533-3.823,8.533-8.533v-24.909
c6.153,4.838,14.157,7.842,25.6,7.842c21.146,0,37.436-15.71,42.496-40.994C261.989,278.203,241.696,261.759,222.07,245.8
53z      M239.111,305.544c-1.28,6.4-6.921,27.273-25.762,27.273c-13.952,0-18.415-4.702-26.505-20.881
c-0.401-0.794-0.913-1.485-1.519-2.091c-0.128-0.145-0.299-0.239-0.435-0.375c-0.606-0.538-1.254-0.973-1.971-1.323
c-0.247-0.119-0.486-0.23-0.751-0.333c-0.939-0.35-1.92-0.597-2.953-0.597s-2.022,0.247-2.953,0.597
c-0.265,0.102-0.512,0.213-0.759,0.333c-0.717,0.35-1.365,0.785-1.954,1.323c-0.154,0.137-0.316,0.23-0.452,0.375
c-0.597,0.614-1.126,1.297-1.519,2.091c-8.09,16.179-12.553,20.881-26.496,20.881c-14.123,0-25.6-11.486-25.6-25.6
c0-21.478,12.467-32.512,29.722-47.778c9.745-8.627,20.514-18.159,30.089-30.942c9.856,12.638,21.495,22.076,32.026,30.61
8     C232.284,276.104,243.045,285.891,239.111,305.544z" />
   </g>
  </g>
 </g>
</svg>
```

Entertainment



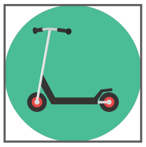
What type of object is shown in the image?

A: Cards B: Dice

C: Coins D: Dominoes

```
<svg id="Layer_1" style="enable-background:new 0 0 496.158 496.158;" version="1.1" viewBox="0 0 496.158 496.158"
x="0px" y="0px" xml:space="preserve">
  <path d="M248.082,0.002C111.07,0.002,0,111.062,0,248.085c0,137,111.07,248.07,248.082,248.07
c137.006,0,248.076-111.07,248.076-248.07C496.158,111.062,385.088,0.002,248.082,0.002z" style="fill:#4ABC96;" />
  <circle cx="378.479" cy="351.559" r="35.37" style="fill:#353432;" />
  <polygon points="162.277,81.768 110.166,87.496 110.151,96.402 159.947,90.671 225.523,96.999   225.539,88.094 "
style="fill:#D8D8D8;" />
  <polygon points="162.277,86.117 110.166,91.844 110.151,96.402 159.947,90.671 225.523,96.999   225.539,92.442 "
style="fill:#C6C5C4;" />
  <path d="M118.683,91.725c-0.884-6.467-5.077-11.07-9.661-10.44c-4.532,0.621-7.388,6.179-6.503,12.653
c0.834,6.081,4.553,10.495,8.845,10.495c0.271,0,0.544-0.018,0.818-0.055C116.714,103.756,119.568,98.198,118.683,91.72
5z" style="fill:#282827;" />
  <g>
   <path d="M140.037,93.276c0,0-3.748-1.126-4.186-4.476c-0.431-3.294,3.059-4.516,3.059-4.516
c-0.19-1.272-1.685-2.633-2.945-2.441l-24.101,3.61c-0.487-0.121-0.993-0.158-1.508-0.088c-2.931,0.402-4.775,3.995-4.203
,8.181
c0.538,3.932,2.942,6.785,5.717,6.785c0.177,0,0.354-0.012,0.53-0.035c0.699-0.096,1.335-0.379,1.893-0.803l23.785-3.566
c0.611-0.091,1.16-0.423,1.529-0.919C139.975,94.51,140.13,93.887,140.037,93.276z" style="fill:#353432;" />
   <path d="M232.928,88.588l-38.029-5.228c-0.61-0.082-1.232,0.083-1.724,0.459
c-0.491,0.375-2.028,2.085-2.624,5.128c-0.729,3.715,1.321,7.405,2.598,7.574l38.03,5.228c0.104,0.013,0.205,0.021,0.308,0.
021
c0.508,0,1.008-0.167,1.417-0.48c0.49-0.375,0.813-0.931,0.894-1.544l1.135-8.54C235.1,89.928,234.204,88.757,232.928,88
.588z" style="fill:#353432;" />
  </g>
  <circle cx="378.639" cy="351.559" r="19.06" style="fill:#EA4949;" />
  <circle cx="117.679" cy="351.559" r="35.37" style="fill:#353432;" />
  <circle cx="117.679" cy="351.559" r="19.06" style="fill:#EA4949;" />
  <polygon points="159.947,90.671 112.498,352.617 121.284,354.064 168.87,90.499 " style="fill:#D8D8D8;" />
  <polygon points="159.947,90.671 112.498,352.617 116.816,353.87 164.403,90.306 " style="fill:#EDEDED;" />
  <g>
   <circle cx="117.679" cy="351.559" r="9.152" style="fill:#D8D8D8;" />
   <polygon points="380.224,347.533 328.426,347.363 328.135,356.685 380.539,356.431  " style="fill:#D8D8D8;" />
   <circle cx="378.479" cy="351.559" r="9.152" style="fill:#D8D8D8;" />
  </g>
  <path d="M241.206,90.157c-1.334-2.642-3.525-4.505-6.169-5.243c-0.8-0.225-1.621-0.337-2.442-0.337
c-4.663,0-8.789,3.547-10.03,8.627c-1.484,6.08,1.696,12.254,7.09,13.765c0.799,0.224,1.621,0.336,2.443,0.336
c4.664,0,8.789-3.547,10.028-8.626C242.838,95.764,242.512,92.737,241.206,90.157z" style="fill:#282827;" />
  <g>
   <polygon points="348.655,306.124 322.353,337.207 336.875,344.935 356.493,315.413 390.288,311.529    390.288,301
" style="fill:#353432;" />
   <rect height="24.723" style="fill:#353432;" width="167.42" x="170.039" y="334.839" />
   <polygon points="139.347,252.484 132.25,292.924 170.035,359.557 190.058,348.055  " style="fill:#353432;" />
  </g>
</svg>
```

Land Crafts



How many wheels does the object have?

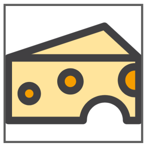
A: One B: Two

C: Three D: Four

Figure 19: SVG examples in our SGP-Bench.

```
<svg id="Layer_1" style="enable-background:new 0 0 276.316 276.316;" version="1.1" viewBox="0 0
276.316 276.316" x="0px" y="0px" xml:space="preserve">
  <g>
    <polygon points="7.5,107.293 268.817,107.293 199.899,43.566  " style="fill:#FFE49C;" />
    <path
d="M7.5,107.293V232.75h149.758c0-24.023,19.475-43.498,43.498-43.498s43.498,19.475,43.498,43.498
h24.563V107.293H7.5z" style="fill:#FFE49C;" />
    <circle cx="49.45" cy="178.833" r="15.965" style="fill:#F59D00;" />
    <circle cx="131.873" cy="154.401" r="18.904" style="fill:#F59D00;" />
    <g>
      <path d="M268.817,177.651V126.24c-2.498-0.767-5.151-1.182-7.9-1.182
c-14.849,0-26.887,12.038-26.887,26.887s12.038,26.887,26.887,26.887C263.666,178.833,266.318,178.418,268
.817,177.651z" style="fill:#F59D00;" />
      <g>
        <path d="M273.908,101.787l-68.917-63.728c-2.004-1.853-4.858-2.471-7.45-1.612L5.287,100.126
C2.11,101.131,0,104.033,0,107.294V232.75c0,4.143,3.358,7.5,7.5,7.5h149.758c4.142,0,7.5-3.357,7.5-7.5
c0-19.85,16.148-35.998,35.998-35.998c19.849,0,35.997,16.148,35.997,35.998c0,4.143,3.358,7.5,7.5,7.5h24.56
3     c4.142,0,7.5-3.357,7.5-7.5v-55.099v-51.41v-18.947C276.316,105.149,275.383,103.157,273.908,101.787z
M261.316,171.329
c-10.801,0.237-19.787-8.51-19.787-19.383c0-10.865,8.978-19.62,19.787-19.383V171.329z
M198.057,52.077l51.602,47.717H53.996     L198.057,52.077z
M251.203,225.25c-3.637-24.578-24.874-43.498-50.447-43.498s-46.811,18.92-50.448,43.498H15V114.794h24
6.316
v2.781c-19.094-0.244-34.787,15.238-34.787,34.372c0,19.054,15.611,34.617,34.787,34.372v38.932h-10.113V
225.25z" style="fill:#414042;" />
        <path d="M49.45,155.368c-12.939,0-23.465,10.526-23.465,23.465s10.526,23.466,23.465,23.466
s23.465-10.526,23.465-23.466C72.915,165.894,62.389,155.368,49.45,155.368z
M49.45,187.299c-4.668,0-8.465-3.798-8.465-8.466
s3.797-8.465,8.465-8.465s8.465,3.797,8.465,8.465S54.118,187.299,49.45,187.299z" style="fill:#414042;" />
        <path d="M131.873,127.997c-14.56,0-26.404,11.845-26.404,26.404s11.845,26.404,26.404,26.404
c14.56,0,26.405-11.845,26.405-26.404C158.278,139.842,146.433,127.997,131.873,127.997z
M131.873,165.805
c-6.288,0-11.404-5.116-11.404-11.404s5.116-11.404,11.404-11.404c6.289,0,11.405,5.116,11.405,11.404
S138.162,165.805,131.873,165.805z" style="fill:#414042;" />
      </g>
    </g>
  </g>
</svg>
```

Dairy



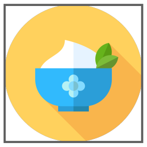
How many circular holes are visible on the object?

A: 2 B: 3

C: 4 D: 5

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <circle cx="256" cy="256" r="256" style="fill:#FFD15D;" />
  <path d="M302.327,507.807c115.54-21.12,203.978-119.794,209.396-240.097L387.01,143l-64.353,59.559
L137.18,342.661L302.327,507.807z" style="fill:#F9B54C;" />
  <path d="M128.434,256c0,0-1.289-24.824,31.892-49.648c33.18-24.824,62.493-33.851,63.783-74.473
c0,0,159.134,30.842,159.458,124.121H128.434z" style="fill:#FFFFFF;" />
  <path d="M256,140.276V256h127.566C383.336,189.526,302.468,154.769,256,140.276z" style="fill:#E6F3FF;" />
  <path d="M393.281,200.932c-8.22,19.834-25.74,32.98-45.411,36.516c-11.404-16.413-14.489-38.098-6.27-57.932
s25.74-32.98,45.411-36.516C398.415,159.413,401.501,181.098,393.281,200.932z" style="fill:#66B31B;" />
  <path d="M387.019,143.01l-39.557,93.806c0.141,0.207,0.267,0.424,0.41,0.631
c19.671-3.536,37.192-16.682,45.411-36.516C401.499,181.103,398.415,159.422,387.019,143.01z" style="fill:#599B13;" /
>
  <g>
    <path d="M392.097,239.14c-17.239,8.601-36.843,7.194-52.248-1.989c1.934-17.834,12.605-34.344,29.843-42.944
c17.237-8.601,36.843-7.194,52.248,1.989C420.007,214.03,409.336,230.54,392.097,239.14z" style="fill:#7CBC39;" />
    <path d="M369.692,194.208c-16.644,8.304-27.396,25.822-29.844,42.944l81.36-41.362
C405.914,186.989,386.662,185.742,369.692,194.208z" style="fill:#7CBC39;" />
  </g>
  <path d="M339.849,237.151C339.761,237.763,339.915,236.535,339.849,237.151
c15.405,9.183,35.011,10.588,52.248,1.989c17.237-8.599,27.91-25.11,29.843-42.944c-0.24-0.143-0.491-0.267-0.733-0.40
7  L339.849,237.151z" style="fill:#65932F;" />
  <rect height="28.444" style="fill:#2B9ED8;" width="113.778" x="199.111" y="369.778" />
<rect height="28.444" style="fill:#2287AF;" width="56.889" x="256" y="369.778" />
  <path d="M297.519,378.828h-83.035c-55.618,0-100.705-45.087-100.705-100.704v-40.657h284.444v40.655
C398.222,333.741,353.135,378.828,297.519,378.828z" style="fill:#31BAFD;" />
  <path d="M256,237.468v141.36h41.517c55.618,0,100.705-45.087,100.705-100.705v-40.655H256z"
style="fill:#2B9ED8;" />
  <circle cx="256" cy="278.928" r="17.965" style="fill:#B5F1F4;" />
  <path d="M273.965,278.921c0-9.921-8.044-17.965-17.965-17.965v35.93
C265.923,296.886,273.965,288.842,273.965,278.921z" style="fill:#84DBFF;" />
  <circle cx="256" cy="328.318" r="17.965" style="fill:#B5F1F4;" />
  <g>
    <path d="M273.965,328.325c0-9.921-8.044-17.965-17.965-17.965v35.93
C265.923,346.29,273.965,338.246,273.965,328.325z" style="fill:#84DBFF;" />
    <circle cx="280.704" cy="303.614" r="17.965" style="fill:#84DBFF;" />
  </g>
  <circle cx="231.296" cy="303.614" r="17.965" style="fill:#B5F1F4;" />
  <circle cx="256" cy="303.614" r="17.965" style="fill:#84DBFF;" />
  <path d="M273.965,303.623c0-9.921-8.044-17.965-17.965-17.965v35.93
C265.923,321.588,273.965,313.544,273.965,303.623z" style="fill:#79C1D1;" />
</svg>
```

Food



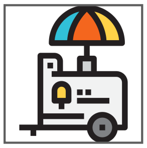
What shape is prominently featured on the front of the bowl?

A: Star B: Circle

C: Triangle D: Flower

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <path d="M362.986,407.436c25.906,0,46.997,21.092,46.997,46.997c0,25.906-21.092,46.997-46.997,46.997
c-25.906,0-46.997-21.092-46.997-46.997C315.989,428.528,337.081,407.436,362.986,407.436z" style="fill:#808285;" />
  <path d="M237.66,318.662v36.553h-26.11h-26.11v-36.553c0-7.21,2.917-13.788,7.629-18.48
c4.692-4.702,11.281-7.629,18.481-7.629C225.962,292.554,237.66,304.251,237.66,318.662z" style="fill:#FFD248;" />
  <path d="M211.551,355.216h26.11v-36.553c0-14.411-11.699-26.108-26.11-26.108
c-7.2,0-13.789,2.926-18.481,7.629c-4.711,4.692-7.629,11.269-7.629,18.48v36.553H211.551z
M133.222,271.665v-62.662v-20.889h20.888
h31.331c11.484,0,20.888,9.404,20.888,20.889v46.997h73.107h52.219h83.551c17.338,0,31.332,13.992,31.332,31.331v167.101h-36.55
3  c0-25.906-21.092-46.997-46.997-46.997c-25.906,0-46.997,21.092-46.997,46.997h-41.775H133.222v-41.775V271.665z"
style="fill:#F1F2F2;" />
  <polygon points="331.655,198.559 331.655,256.001 279.436,256.001 279.436,198.559 305.545,198.559 " style="fill:#D1D3D4;" />
  <path d="M436.093,141.118h-67.885c0-78.33-62.663-130.548-62.663-130.548  C377.612,10.571,436.093,69.052,436.093,141.118z"
style="fill:#FFD248;" />
  <path d="M242.882,141.118c0-78.33,62.663-130.548,62.663-130.548s62.663,52.218,62.663,130.548h-62.663  H242.882z"
style="fill:#F4661E;" />
  <path d="M305.545,10.571c0,0-62.663,52.218-62.663,130.548h-67.885  C174.997,69.052,233.479,10.571,305.545,10.571z"
style="fill:#31C0D8;" />
  <g>
    <path d="M415.205,245.43h-72.98v-46.87c0-5.838-4.734-10.571-10.571-10.571h-15.539v-36.3h119.977
c5.837,0,10.571-4.732,10.571-10.571C446.664,63.306,383.358,0,305.545,0S164.427,63.306,164.427,141.118
c0,5.838,4.734,10.571,10.571,10.571h119.977v36.3h-15.539c-5.837,0-10.571,4.732-10.571,10.571v46.87H216.9v-36.427
c0-17.346-14.113-31.459-31.458-31.459h-52.219c-5.837,0-10.571,4.732-10.571,10.571v255.747H54.893v21.141h41.648v20.761h21.1
41
v-20.761h188.725c4.979,26.71,28.447,46.997,56.577,46.997s51.599-20.287,56.577-46.997h26.973c5.838,0,10.571-4.734,10.571-10.5
71   V287.331C457.108,264.227,438.31,245.43,415.205,245.43z
M305.545,25.104c14.511,14.679,47.59,53.472,51.672,105.444H253.874   C257.955,78.576,291.034,39.784,305.545,25.104z
M425.06,130.548h-46.64c-3.142-47.255-26.711-84.246-44.844-106.096   C382.935,36.311,420.519,78.762,425.06,130.548z
M277.513,24.452c-18.133,21.85-41.702,58.841-44.844,106.096h-46.64   C190.57,78.762,228.155,36.311,277.513,24.452z
M290.006,209.13h31.078v36.3h-31.078V209.13z M362.986,490.859
c-20.085,0-36.427-16.341-36.427-36.427c0-20.085,16.341-36.427,36.427-36.427s36.427,16.341,36.427,36.427
C399.413,474.517,383.072,490.859,362.986,490.859z M435.966,443.861h-16.403c-4.979-26.71-28.447-46.997-56.577-46.997
c-28.131,0-51.6,20.287-56.577,46.997H143.793v-20.634h151.309v-21.141H143.793V198.685h41.647c5.689,0,10.317,4.629,10.317,1
0.318   v36.427h-20.761v21.141h31.331h208.878c11.447,0,20.761,9.314,20.761,20.761V443.861z" style="fill:#231F20;" />
    <rect height="21.141" style="fill:#231F20;" width="20.888" x="352.542" y="443.862" />
    <path d="M211.551,281.983c-9.694,0-19.158,3.912-25.94,10.709c-6.926,6.897-10.74,16.12-10.74,25.97v36.553
c0,5.837,4.732,10.571,10.571,10.571h15.539v20.762h21.141v-20.762h15.539c5.838,0,10.571-4.734,10.571-10.571v-36.553
C248.231,298.438,231.777,281.983,211.551,281.983z M227.09,344.645h-31.078v-25.983c0-4.186,1.605-8.089,4.541-11.014
c2.869-2.876,6.877-4.524,10.998-4.524c8.568,0,15.539,6.97,15.539,15.538L227.09,344.645L227.09,344.645z" style="fill:#231F20;" /
>
    <rect height="21.141" style="fill:#231F20;" width="20.888" x="268.992" y="313.315" />
    <rect height="21.141" style="fill:#231F20;" width="120.104" x="268.992" y="344.646" />
    <rect height="21.141" style="fill:#231F20;" width="20.888" x="300.323" y="313.315" />
    <rect height="21.141" style="fill:#231F20;" width="20.889" x="159.332" y="214.098" />
  </g>
</svg>
```

Food



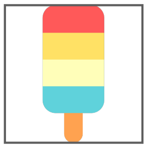
What is the object in the image?

A: popsicle cart B: truck

C: bicycle D: train

```
<svg id="Layer_1" style="enable-background:new 0 0 371.671 371.671;"
version="1.1" viewBox="0 0 371.671 371.671" x="0px" y="0px"
xml:space="preserve">
  <g id="XMLID_1441_">
    <path
d="M185.835,371.671L185.835,371.671c-13.738,0-24.978-11.24-24.978-24.978
v-95.902
c0-13.738,11.24-24.978,24.978-24.978l0,0c13.738,0,24.978,11.24,24.978,24.978
v95.902   C210.813,360.431,199.574,371.671,185.835,371.671z"
id="XMLID_583_" style="fill:#FFA250;" />
    <path
d="M268.836,35.02v221.272c0,19.33-15.67,35-35,35h-96c-19.33,0-35-15.67-35
-35V35.02
c0-19.33,15.67-35.02,35-35.02h96C253.165,0,268.836,15.69,268.836,35.02z"
id="XMLID_1442_" style="fill:#5FD2DB;" />
    <rect height="72.13" id="XMLID_1443_" style="fill:#FFFEB9;" width="166"
x="102.835" y="147.032" />
    <rect height="72.12" id="XMLID_1444_" style="fill:#FFE165;" width="166"
x="102.835" y="74.912" />
    <path
d="M268.836,35.02v39.892h-166V35.02c0-19.33,15.67-35.02,35-35.02h96
C253.165,0,268.836,15.69,268.836,35.02z" id="XMLID_1445_"
style="fill:#FF5959;" />
  </g>
</svg>
```

Food



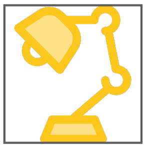
How many distinct color sections are there in the object's main body?

A: 2 B: 3

C: 4 D: 5

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px"
y="0px" xml:space="preserve">
  <g>
    <polygon points="367.819,496.989 149.426,496.989 179.462,424.937 337.781,424.937  "
style="fill:#FEE187;" />
    <path d="M207.987,239.881L53.56,85.456l81.201-53.188c24.429-16.001,56.734-12.671,77.384,7.979
l41.051,41.051c20.649,20.649,23.982,52.955,7.979,77.384L207.987,239.881z" style="fill:#FEE187;" />
  </g>
  <path d="M429.072,224.281L405.166,103.68c16.581-9.631,27.761-27.575,27.761-48.091
C432.928,24.938,407.991,0,377.339,0c-25.451,0-46.947,17.199-53.513,40.58h-90.122l-10.948-10.948
c-14.481-14.481-33.74-22.457-54.232-22.457c-14.951,0-29.469,4.336-41.988,12.536L45.335,72.899
c-3.779,2.475-6.242,6.523-6.706,11.017c-0.464,4.494,1.123,8.957,4.317,12.153l36.882,36.882
c-8.557,10.265-13.22,23.072-13.22,36.6c0,15.3,5.959,29.686,16.779,40.505c10.819,10.82,25.205,16.779,40.506,1
6.779
c13.528,0,26.335-4.663,36.6-13.22l36.882,36.882c2.829,2.829,6.654,4.396,10.613,4.396c0.512,0,1.027-0.027,1.5
4-0.08
c4.494-0.464,8.54-2.927,11.015-6.706l53.188-81.201c19.791-30.215,15.617-70.681-9.921-96.22l-0.087-0.087h73
.037
c8.288,0,15.009-6.721,15.009-15.009c0-14.101,11.471-25.571,25.571-25.571c14.101,0,25.571,11.471,25.571,25.
571
S391.439,81.16,377.339,81.16c-8.288,0-15.009,6.721-15.009,15.009c0,7.848,6.029,14.283,13.708,14.943l25.634,
129.321
c1.06,5.343,4.834,9.456,9.606,11.192c1.99,0.974,4.219,1.537,6.583,1.537c14.101,0,25.571,11.471,25.571,25.571
s-11.471,25.569-25.571,25.569s-25.571-11.471-25.571-25.569c0-8.288-6.721-15.009-15.009-15.009
c-8.288,0-15.009,6.721-15.009,15.009c0,9.856,2.591,19.114,7.108,27.15L246.822,409.928h-67.358
c-6.058,0-11.522,3.643-13.853,9.234l-30.038,72.053c-1.932,4.632-1.42,9.922,1.364,14.1c2.783,4.177,7.47,6.685,
12.489,6.685
h218.393c0.012,0.001,0.023,0.001,0.03,0c8.289,0,15.009-6.721,15.009-15.009c0-2.391-0.557-4.65-1.552-6.655l-
29.67-71.173
c-2.332-5.592-7.796-9.234-13.853-9.234h-44.58l97.463-82.739c8.046,4.534,17.32,7.132,27.195,7.132
c30.652,0,55.589-24.936,55.589-55.588C473.45,251.923,454.369,229.483,429.072,224.281z M123.893,196.815
c-7.284,0-14.131-2.837-19.281-7.986s-7.985-11.995-7.985-19.279c0-5.492,1.612-10.734,4.612-15.189l37.842,37.
842  C134.626,195.203,129.383,196.815,123.893,196.815z M248.618,150.457l-43.073,65.759l-34.254-34.254
c-0.005-0.005-0.007-0.009-0.012-0.014l-59.787-59.787c-0.003-0.003-0.006-0.006-0.011-0.009L77.227,87.896l65
.758-43.073  c7.617-4.989,16.448-7.628,25.539-7.628c12.473,0,24.195,4.852,33.006,13.664l41.053,41.053
C258.121,107.451,260.662,132.074,248.618,150.457z
M345.301,481.98H171.944l17.523-42.034h138.31L345.301,481.98z" style="fill:#FFC61B;" />
</svg>
```

Furniture



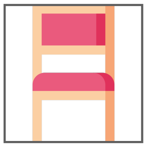
How many joints does the arm of the object have?

A: One B: Two

C: Three D: Four

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1"
viewBox="0 0 512 512" x="0px" y="0px" xml:space="preserve">
  <polygon points="407.912,0 407.912,320.703 374.154,320.703
374.154,185.67 340.396,92.835 374.154,0 " style="fill:#F7A676;" />
  <polygon points="374.154,0 374.154,33.758 340.396,92.835 374.154,151.912
374.154,185.67 137.846,185.67   137.846,320.703 104.088,320.703 104.088,0 "
style="fill:#FBCFA3;" />
  <polygon points="374.154,33.758 374.154,151.912 340.396,151.912
306.637,92.835 340.396,33.758 " style="fill:#E0315B;" />
  <rect height="118.154" style="fill:#EA5A7D;" width="202.549" x="137.846"
y="33.758" />
  <path d="M362.901,253.187h-33.758v90.022l78.769-22.505v-22.505
C407.912,273.442,387.657,253.187,362.901,253.187z" style="fill:#E0315B;" /
>
  <path
d="M374.154,298.198v33.758h-247.56l-22.506-11.253v-22.505c0-24.756,20.2
55-45.011,45.011-45.011
h180.044C353.899,253.187,374.154,273.442,374.154,298.198z"
style="fill:#EA5A7D;" />
  <polygon points="104.088,320.703 104.088,512 137.846,512
137.846,354.462 385.407,354.462 374.154,320.703   " style="fill:#FBCFA3;" /
>
  <rect height="191.297" style="fill:#F7A676;" width="33.758" x="374.154"
y="320.703" />
</svg>
```

Furniture



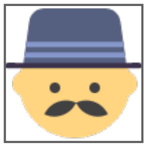
How many visible legs does the object in the image have?

A: Two B: Three

C: Four D: Five

```
<svg id="Capa_1" style="enable-background:new 0 0 52 52;" version="1.1" viewBox="0 0
52 52" x="0px" y="0px" xml:space="preserve">
  <g>
    <path
d="M46,23.122H7L9.339,6.347C9.718,3.57,12.089,1.5,14.892,1.5h23.216c2.803,0,5.174,2.0
7,5.553,4.847   L46,23.122z" style="fill:#556080;" />
    <path
d="M44.745,14.122H8.255l-0.274,1.967c0.055,0.009,0.105,0.033,0.162,0.033h36.715
c0.057,0,0.107-0.023,0.162-0.033L44.745,14.122z" style="fill:#8697CB;" />
    <path
d="M45.303,18.122H7.697l-0.274,1.965c0.059,0.011,0.112,0.035,0.174,0.035h37.806
c0.062,0,0.115-0.024,0.174-0.035L45.303,18.122z" style="fill:#8697CB;" />
    <path
d="M50,30c0-1.898-1.337-3.839-3-4.5c-0.277-1.567-1-2.378-1-2.378H7c0,0-0.723,0.811-1,
2.378
c-1.663,0.661-3,2.602-3,4.5c0,1.978,1.284,3.639,3.058,4.242C8.21,43.552,16.536,50.5,26.5,
50.5   c9.964,0,18.29-6.947,20.442-16.258C48.716,33.639,50,31.978,50,30z"
style="fill:#FFD581;" />
    <circle cx="19" cy="31.5" r="2" style="fill:#414141;" />
    <circle cx="34" cy="31.5" r="2" style="fill:#414141;" />
    <path
d="M26.965,37.159C22.458,35.052,17.107,36.994,15,41.5C19.506,43.607,24.858,41.665,26.
965,37.159z" style="fill:#414141;" />
    <path
d="M27,37.159c4.506-2.107,9.858-0.165,11.965,4.341C34.458,43.607,29.107,41.665,27,37.
159z" style="fill:#414141;" />
    <path
d="M51,24.122H1c-0.552,0-1-0.447-1-1s0.448-1,1-1h50c0.552,0,1,0.447,1,1S51.552,24.122
,51,24.122z" style="fill:#3D324C;" />
  </g>
</svg>
```

Human



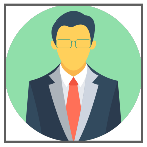
What facial feature is prominent on the object?

A: Glasses B: Beard

C: Mustache D: Scar

```
<svg id="Layer_1" style="enable-background:new 0 0 508 508;" version="1.1" viewBox="0 0 508 508" x="0px" y="0px" xml:space="preserve">
  <circle cx="254" cy="254" r="254" style="fill:#90DFAA;" />
  <path d="M305.2,245.6c0,34.4-23.2,45.6-51.2,45.6s-51.2-11.2-51.2-45.6S225.6,166,254,166  S305.2,211.2,305.2,245.6z" style="fill:#F9B54C;" />
  <path d="M307.6,238l-2.8,142l-16.4,46.4L259.6,508c-1.6,0-2.8,0-4.4,0c-0.4,0-1.2,0-2,0c-0.4,0-1.2,0-2,0
c-1.6,0-3.2,0-4.4,0L218,428l-24-67.2V252.4l8.4-20l0.4,0.4l2.8,2.8l31.6,32.8l5.6,5.6l10-5.6l0.4-0.4l8.8-4.8l0,0l38.8-22l3.6-2  l0,0L307.6,238z"
style="fill:#E6E9EE;" />
  <polygon points="236.4,268.4 226,282.4 239.2,300 268.8,300 282,282.4 271.6,268.4 " style="fill:#F1543F;" />
  <path d="M295.6,470l-14.4,16.4l0,0l-18.8,21.2c-0.8,0-1.6,0-2.4,0c-1.6,0-2.8,0-4.4,0c-0.4,0-1.2,0-2,0
c-0.4,0-1.2,0-2,0c-1.6,0-3.2,0-4.4,0c-0.8,0-1.6,0-2,0l-18.8-21.2l0,0L212,470l2-11.6l4.8-30.8l6.8-44l0,0l13.2-83.6h29.6
l13.6,83.6l0,0l0,0l6.8,42.8l5.2,32L295.6,470z" style="fill:#FF7058;" />
  <path d="M148.4,411.2c-3.6,2.8-7.2,30.8-11.2,68.4c-0.4,0-0.8-0.4-0.8-0.4c-21.6-11.2-41.2-25.6-58.8-42.4
c6.4-84,12.4-154,12.4-154l0,0c0,0,0,0,0,0,0.4L148.4,411.2z" style="fill:#324A5E;" />
  <path d="M252,508c-1.6,0-3.2,0-4.4,0c-0.8,0-1.6,0-2,0c-2.8,0-5.2,0-8-0.4c-36-2.4-70-12-100-27.6
c-0.4,0-0.8-0.4-0.8-0.4c-15.2-100.4-44-189.6-46-196c0,0,0,0,0-0.4l0,0c74.8-16,113.2-63.6,117.2-69.2l0,0c0-0.4,0.4-0.4,0.4-0.4
c0,0.4,0,0.8-0.4,1.2c-1.2,5.2-2,12.4-2,21.2c0,8.4,0.4,18,1.6,28.8l0,0c0,2,0.4,4,0.4,6.4l0,0c3.2,33.2,10.4,74,17.6,112.4l0,0
c12.4,65.2,26,123.2,26,123.2C252,507.2,252,507.6,252,508z" style="fill:#2B3B4E;" />
  <path d="M252,508c-1.6,0-3.2,0-4.4,0c-0.8,0-1.6,0-2,0c-2.8,0-5.2,0-8-0.4c-3.6-6.8-7.2-13.6-10.8-20.8l0,0
c-4.4-8.8-8.4-18.4-12.4-28c-31.6-74.8-52.8-161.6-52.8-161.6c6.4,0.8,23.6-7.2,23.6-7.2l-23.6-27.6c25.2-6.4,42.8-40,46.8-47.6
c0.4-0.8,0.8-1.2,0.8-1.6c0,0.4,0,0.8-0.4,1.2c-1.2,5.2-2,12.4-2,21.2c0,8.4,0.4,18,1.6,28.8l0,0c0,2,0.4,4,0.4,6.4l0,0
c3.2,33.2,10.4,74,17.6,112.4l0,0c12.4,65.2,26,123.2,26,123.2C252,507.2,252,507.6,252,508z" style="fill:#ACB3BA;" />
  <path d="M430.4,436.8c-17.2,16.8-37.2,31.2-58.8,42.4c-0.4,0-0.8,0.4-0.8,0.4c-3.6-38-7.6-65.6-11.2-68.4
l58-128c0,0,0,0,0-0.4l0,0C417.6,282.8,424,353.2,430.4,436.8z" style="fill:#2B3B4E;" />
  <path d="M417.6,282.8L417.6,282.8C417.6,283.2,417.6,283.2,417.6,282.8c-2,6.8-30.8,96-46,196.4
c-0.4,0-0.8,0.4-0.8,0.4c-30.4,15.6-64.4,25.6-100,27.6c-2.8,0-5.2,0.4-8,0.4c-0.8,0-1.6,0-2.4,0c-1.6,0-2.8,0-4.4,0
c0-0.4,0-0.8,0-0.8s13.6-58,26-123.2l0,0l0,0c7.2-38.4,14-79.6,17.6-112.4l0,0c0-2,0.4-4,0.4-6l0,0c0.8-8.4,1.2-16.4,1.6-23.2
c0.4-11.6-0.4-20.8-2-27.2l0,0c0-0.4,0-0.8-0.4-1.2l0,0c0,0,0,0,0,0.4,0.4l0,0C303.2,218.4,341.6,266.8,417.6,282.8z" style="fill:#324A5E;" />
  <path d="M346.8,297.2c0,0-21.2,86.4-52.8,161.6c-4,9.6-8.4,18.8-12.4,28l0,0c-3.6,7.2-7.2,14.4-10.8,20.8
c-2.8,0-5.2,0.4-8,0.4c-0.8,0-1.6,0-2.4,0c-1.6,0-2.8,0-4.4,0c0-0.4,0-0.8,0-0.8s13.6-58,26-123.2l0,0l0,0
c7.2-38.4,14-79.6,17.6-112.4l0,0c0-2,0.4-4,0.4-6l0,0c0.8-8.4,1.2-16.4,1.6-23.2c0.4-11.6-0.4-20.8-2-27.2l0,0c0-0.4,0-0.8-0.4-1.2
l0,0c0,0,0,0.4,0.4,0.4,1.2l0,0c3.6,7.2,21.6,41.2,46.8,47.6L323.2,290C323.2,290,340.4,298,346.8,297.2z" style="fill:#CED5E0;" />
  <g>
    <path d="M299.2,213.2c0,0,11.2,25.2-45.2,54.8l25.6,38C279.6,306.4,320,248.8,299.2,213.2z" style="fill:#FFFFFF;" />
    <path d="M208.8,213.2c0,0-11.2,25.2,45.2,54.8l-25.6,38C228.4,306.4,188,248.8,208.8,213.2z" style="fill:#FFFFFF;" />
  </g>
  <g>
    <path d="M320.8,130.4c0,49.6-30,114.4-66.8,114.4s-66.8-64.8-66.8-114.4s30-65.2,66.8-65.2   S320.8,80.8,320.8,130.4z" style="fill:#FFD05B;" /
>
    <path d="M335.6,154.4c-4,9.2-11.6,14.8-16.8,12.8c-5.2-2-6.4-10.8-2.4-20s11.6-14.8,16.8-12.8   C338.8,136,339.6,145.2,335.6,154.4z"
style="fill:#FFD05B;" />
    <path d="M172.4,154.4c4,9.2,11.6,14.8,16.8,12.8s6.4-10.8,2.4-20s-11.6-14.8-16.8-12.8   C169.2,136,168.4,145.2,172.4,154.4z"
style="fill:#FFD05B;" />
  </g>
  <path d="M253.6,82.8c19.6-10.8,65.2-26,65.2,72c0,0,44-112.4-23.2-114c0,0-47.2-48-108.4,22l9.6-4.8
c0,0-49.2,28.8-7.2,96.8c0,0-7.2-98.4,47.2-71.6C242,86,248.4,85.6,253.6,82.8z" style="fill:#324A5E;" />
  <g>
    <path d="M309.6,161.6h-40c-4.4,0-8-3.6-8-8v-14.8   c0-4.4,3.6-8,8-8h40c4.4,0,8,3.6,8,8v14.8C317.6,158,314,161.6,309.6,161.6z"
style="fill:none;stroke:#2C9984;stroke-width:1.549;stroke-miterlimit:10;" />
    <path d="M238.4,161.6h-40c-4.4,0-8-3.6-8-8v-14.8   c0-4.4,3.6-8,8-8h40c4.4,0,8,3.6,8,8v14.8C246,158,242.4,161.6,238.4,161.6z"
style="fill:none;stroke:#2C9984;stroke-width:1.549;stroke-miterlimit:10;" />
    <path d="M246.4,143.2   c0-4.4,3.6-7.6,7.6-7.6s7.6,3.6,7.6,7.6" style="fill:none;stroke:#2C9984;stroke-width:1.549;stroke-linecap:round;stroke-
miterlimit:10;" />
  </g>
</svg>
```

Human

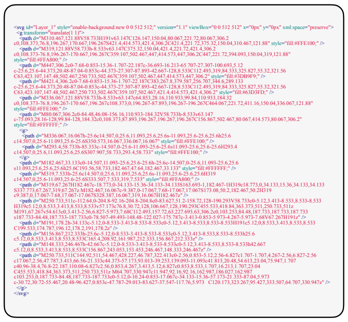


What color is the tie in the image?

A: Blue B: Green

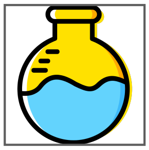
C: Yellow D: Red

Figure 20: SVG examples in our SGP-Bench.



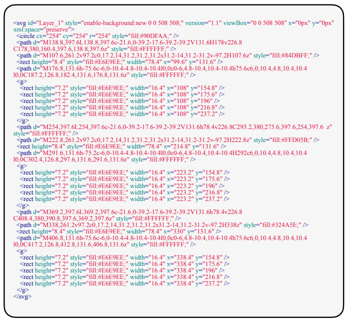
```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px" xml:space="preserve">
  <g transform="translate(1 1)">
    <path d="M310.467,121.88V58.733H191v63.147C126.147,150.04,80.067,221.72,80.067,306.2
c0,108.373,76.8,196.267,170.667,196.267S421.4,414.573,421.4,306.2C421.4,221.72,375.32,150.04,310.467,121.88" style="fill:#FFE100;" />
    <path d="M319,121.88V58.733h-8.533v63.147C375.32,150.04,421.4,221.72,421.4,306.2
c0,108.373-76.8,196.267-170.667,196.267C359.107,502.467,447,414.573,447,306.2C447,221.72,394.093,150.04,319,121.88"
style="fill:#FFA800;" />
    <path d="M447,306.2c0-7.68-0.853-15.36-1.707-22.187c-36.693-16.213-65.707-27.307-100.693,5.12
c-25.6,25.6-44.373,20.48-87.04-0.853c-44.373-27.307-87.893-42.667-128,8.533C112.493,319,84.333,325.827,55.32,321.56
C63,423.107,147.48,502.467,250.733,502.467C359.107,502.467,447,414.573,447,306.2" style="fill:#3DB9F9;" />
    <path d="M421.4,306.2c0-7.68-0.853-15.36-1.707-22.187C383,267.8,379.587,256.707,344.6,289.133
c-25.6,25.6-44.373,20.48-87.04-0.853c-44.373-27.307-87.893-42.667-128,8.533C112.493,319,84.333,325.827,55.32,321.56
C63,423.107,147.48,502.467,250.733,502.467C359.107,502.467,421.4,414.573,421.4,306.2" style="fill:#63D3FD;" />
    <path d="M336.067,121.88V58.733h-8.533v63.147c64.853,28.16,110.933,99.84,110.933,184.32
c0,108.373-76.8,196.267-170.667,196.267c108.373,0,196.267-87.893,196.267-196.267C464.067,221.72,411.16,150.04,336.067,121.88"
style="fill:#FFE100;" />
    <path d="M80.067,306.2c0-84.48,46.08-156.16,110.933-184.32V58.733h-8.533v63.147
c-75.093,28.16-128,99.84-128,184.32c0,108.373,87.893,196.267,196.267,196.267C156.867,502.467,80.067,414.573,80.067,306.2"
style="fill:#FFFFFF;" />
    <g>
      <path d="M336.067,16.067h-25.6c14.507,0,25.6,11.093,25.6,25.6s-11.093,25.6-25.6,25.6h25.6
c14.507,0,25.6-11.093,25.6-25.6S350.573,16.067,336.067,16.067" style="fill:#FFE100;" />
      <path d="M293.4,58.733h-85.333c-14.507,0-25.6-11.093-25.6-25.6s11.093-25.6,25.6-25.6H293.4
c14.507,0,25.6,11.093,25.6,25.6S307.907,58.733,293.4,58.733" style="fill:#FFE100;" />
    </g>
    <path d="M182.467,33.133c0-14.507,11.093-25.6,25.6-25.6h-25.6c-14.507,0-25.6,11.093-25.6,25.6
s11.093,25.6,25.6,25.6h25.6C193.56,58.733,182.467,47.64,182.467,33.133" style="fill:#FFFFFF;" />
    <path d="M319,7.533h-25.6c14.507,0,25.6,11.093,25.6,25.6s-11.093,25.6-25.6,25.6H319
c14.507,0,25.6-11.093,25.6-25.6S333.507,7.533,319,7.533" style="fill:#FFA800;" />
    <path d="M319,67.267H182.467c-18.773,0-34.133-15.36-34.133-34.133S163.693-1,182.467-1H319c18.773,0,34.133,15.36,34.133,34.133
S337.773,67.267,319,67.267z M182.467,16.067c-9.387,0-17.067,7.68-17.067,17.067S173.08,50.2,182.467,50.2H319
c9.387,0,17.067-7.68,17.067-17.067S328.387,16.067,319,16.067H182.467z" />
    <path d="M250.733,511c-112.64,0-204.8-92.16-204.8-204.8c0-83.627,51.2-158.72,128-190.293V58.733c0-5.12,3.413-8.533,8.533-8.533
H319c5.12,0,8.533,3.413,8.533,8.533v57.173c76.8,30.72,128,106.667,128,190.293C455.533,418.84,363.373,511,250.733,511z
M191,67.267v54.613c0,3.413-2.56,6.827-5.973,7.68C112.493,157.72,63,227.693,63,306.2c0,103.253,84.48,187.733,187.733,187.733
s187.733-84.48,187.733-187.733c0-78.507-49.493-148.48-122.027-175.787c-3.413-0.853-5.973-4.267-5.973-7.68V67.267H191z" />
    <path d="M191,178.2h-34.133c-5.12,0-8.533-3.413-8.533-8.533c0-5.12,3.413-8.533,8.533-8.533H191c5.12,0,8.533,3.413,8.533,8.533
C199.533,174.787,196.12,178.2,191,178.2z" />
    <path d="M156.867,212.333h-25.6c-5.12,0-8.533-3.413-8.533-8.533c0-5.12,3.413-8.533,8.533-8.533h25.6
c5.12,0,8.533,3.413,8.533,8.533C165.4,208.92,161.987,212.333,156.867,212.333z" />
    <path d="M148.333,246.467h-42.667c-5.12,0-8.533-3.413-8.533-8.533c0-5.12,3.413-8.533,8.533-8.533h42.667
c5.12,0,8.533,3.413,8.533,8.533C156.867,243.053,153.453,246.467,148.333,246.467z" />
    <path d="M250.733,511C144.92,511,54.467,428.227,46.787,322.413c0-2.56,0.853-5.12,2.56-6.827c1.707-1.707,4.267-2.56,6.827-2.56
c17.067,2.56,47.787,3.413,66.56-21.333c44.373-57.173,93.013-39.253,139.093-11.093c41.813,20.48,54.613,23.04,75.947,1.707
c40.96-38.4,76.8-22.187,110.08-6.827c2.56,0.853,4.267,3.413,5.12,6.827c0.853,8.533,1.707,16.213,1.707,23.04
C455.533,418.84,363.373,511,250.733,511z M64.707,330.947c11.947,92.16,92.16,162.987,186.027,162.987
c103.253,0,187.733-84.48,187.733-187.733c0-5.12,0-10.24-0.853-17.067c-34.133-15.36-57.173-21.333-87.04,5.973
c-30.72,30.72-55.467,20.48-96.427,0.853c-47.787-29.013-83.627-37.547-117.76,5.973   C120.173,323.267,95.427,333.507,64.707,330.947z" />
  </g>
</svg>
```

Science



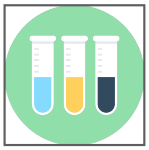
What is the object in the image?

A: Flask   B: Beaker
C: Tube   D: Pipette



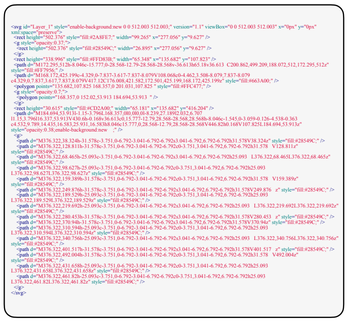
```
<svg id="Layer_1" style="enable-background:new 0 0 508 508;" version="1.1" viewBox="0 0 508 508" x="0px" y="0px"
xml:space="preserve">
  <circle cx="254" cy="254" r="254" style="fill:#90DFAA;" />
  <path d="M138.8,397.6L138.8,397.6c-21.6,0-39.2-17.6-39.2-39.2V131.6H178v226.8
C178,380,160.4,397.6,138.8,397.6z" style="fill:#FFFFFF;" />
  <path d="M107.6,261.2v97.2c0,17.2,14,31.2,31.2,31.2s31.2-14,31.2-31.2v-97.2H107.6z" style="fill:#84DBFF;" />
  <rect height="8.4" style="fill:#E6E9EE;" width="78.4" x="99.6" y="131.6" />
  <path d="M176.8,131.6h-75.6c-6,0-10.4-4.8-10.4-10.4l0,0c0-6,4.8-10.4,10.4-10.4h75.6c6,0,10.4,4.8,10.4,10.4
l0,0C187.2,126.8,182.4,131.6,176.8,131.6z" style="fill:#FFFFFF;" />
  <g>
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="108" y="154.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="108" y="175.6" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="108" y="196" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="108" y="216.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="108" y="237.2" />
  </g>
  <path d="M254,397.6L254,397.6c-21.6,0-39.2-17.6-39.2-39.2V131.6h78.4v226.8C293.2,380,275.6,397.6,254,397.6  z"
style="fill:#FFFFFF;" />
  <path d="M222.8,261.2v97.2c0,17.2,14,31.2,31.2,31.2s31.2-14,31.2-31.2v-97.2H222.8z" style="fill:#FFD05B;" />
  <rect height="8.4" style="fill:#E6E9EE;" width="78.4" x="214.8" y="131.6" />
  <path d="M291.6,131.6h-75.2c-6,0-10.4-4.8-10.4-10.4l0,0c0-6,4.8-10.4,10.4-10.4H292c6,0,10.4,4.8,10.4,10.4
l0,0C302.4,126.8,297.6,131.6,291.6,131.6z" style="fill:#FFFFFF;" />
  <g>
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="223.2" y="154.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="223.2" y="175.6" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="223.2" y="196" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="223.2" y="216.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="223.2" y="237.2" />
  </g>
  <path d="M369.2,397.6L369.2,397.6c-21.6,0-39.2-17.6-39.2-39.2V131.6h78.4v226.8
C408.4,380,390.8,397.6,369.2,397.6z" style="fill:#FFFFFF;" />
  <path d="M338,261.2v97.2c0,17.2,14,31.2,31.2,31.2s31.2-14,31.2-31.2v-97.2H338z" style="fill:#324A5E;" />
  <rect height="8.4" style="fill:#E6E9EE;" width="78.4" x="330" y="131.6" />
  <path d="M406.8,131.6h-75.6c-6,0-10.4-4.8-10.4-10.4l0,0c0-6,4.8-10.4,10.4-10.4h75.6c6,0,10.4,4.8,10.4,10.4
l0,0C417.2,126.8,412.8,131.6,406.8,131.6z" style="fill:#FFFFFF;" />
  <g>
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="338.4" y="154.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="338.4" y="175.6" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="338.4" y="196" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="338.4" y="216.8" />
    <rect height="7.2" style="fill:#E6E9EE;" width="16.4" x="338.4" y="237.2" />
  </g>
</svg>
```

Science



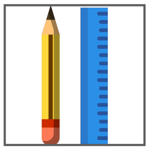
What is the color of the liquid in the middle test tube?

A: Blue   B: Yellow
C: Green   D: Red



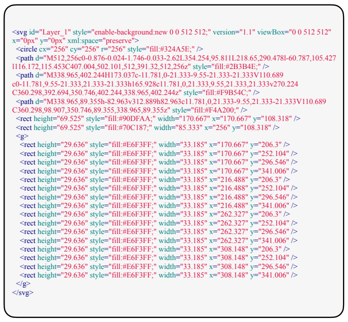
```
<svg id="Layer_1" style="enable-background:new 0 0 512.003 512.003;" version="1.1" viewBox="0 0 512.003 512.003" x="0px" y="0px"
xml:space="preserve">
  <rect height="502.376" style="fill:#2A8FE7;" width="99.265" x="277.056" y="9.627" />
  <g style="opacity:0.37;">
    <rect height="502.376" style="fill:#28549C;" width="26.895" x="277.056" y="9.627" />
  </g>
  <rect height="338.996" style="fill:#FFD83B;" width="65.348" x="135.682" y="107.823" />
  <path d="M172.295,512h-8.046c-15.777,0-28.568-12.79-28.568-28.568v-36.613h65.18v36.613  C200.862,499.209,188.072,512,172.295,512z"
style="fill:#FF7956;" />
  <path d="M168.172,425.199c-4.329,0-7.837-3.617-7.837-8.079V108.068c0-4.462,3.508-8.079,7.837-8.079
c4.329,0,7.837,3.617,7.837,8.079V417.12C176.008,421.582,172.501,425.199,168.172,425.199z" style="fill:#663A00;" />
  <polygon points="135.682,107.825 168.357,0 201.031,107.825 " style="fill:#FFC477;" />
  <g style="opacity:0.7;">
    <polygon points="168.357,0 152.02,53.913 184.694,53.913  " />
  </g>
  <rect height="30.615" style="fill:#CD2A00;" width="65.181" x="135.682" y="416.204" />
  <path d="M184.694,53.913l-1.15-3.796L168.357,0l0,0l0,0l-8.239,27.189l2.032,6.707
l1.15,3.796l16.337,53.913V430.6h-0.168v36.613c0,15.777-12.79,28.568-28.568,28.568h-8.046c-1.545,0-3.059-0.126-4.538-0.363
c4.532,9.789,14.435,16.583,25.931,16.583h8.046c15.777,0,28.568-12.79,28.568-28.568V446.82h0.168V107.825L184.694,53.913z"
style="opacity:0.38;enable-background:new    ;" />
  <g>
    <path d="M376.322,38.324h-31.578c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h31.578V38.324z" style="fill:#28549C;" />
    <path d="M376.322,128.811h-31.578c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h31.578   V128.811z"
style="fill:#28549C;" />
    <path d="M376.322,68.465h-25.093c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h25.093   L376.322,68.465L376.322,68.465z"
style="fill:#28549C;" />
    <path d="M376.322,98.627h-25.093c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h25.093
L376.322,98.627L376.322,98.627z" style="fill:#28549C;" />
    <path d="M376.322,159.389h-31.578c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h31.578   V159.389z"
style="fill:#28549C;" />
    <path d="M376.322,249.876h-31.578c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h31.578V249.876   z" style="fill:#28549C;" />
    <path d="M376.322,189.529h-25.093c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h25.093
L376.322,189.529L376.322,189.529z" style="fill:#28549C;" />
    <path d="M376.322,219.692h-25.093c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h25.093   L376.322,219.692L376.322,219.692z"
style="fill:#28549C;" />
    <path d="M376.322,280.453h-31.578c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h31.578V280.453   z" style="fill:#28549C;" />
    <path d="M376.322,370.94h-31.578c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h31.578V370.94z" style="fill:#28549C;" />
    <path d="M376.322,310.594h-25.093c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h25.093
L376.322,310.594L376.322,310.594z" style="fill:#28549C;" />
    <path d="M376.322,340.756h-25.093c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h25.093   L376.322,340.756L376.322,340.756z"
style="fill:#28549C;" />
    <path d="M376.322,401.517h-31.578c-3.751,0-6.792-3.041-6.792-6.792s3.041-6.792,6.792-6.792h31.578V401.517   z" style="fill:#28549C;" />
    <path d="M376.322,492.004h-31.578c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h31.578   V492.004z"
style="fill:#28549C;" />
    <path d="M376.322,431.658h-25.093c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h25.093
L376.322,431.658L376.322,431.658z" style="fill:#28549C;" />
    <path d="M376.322,461.82h-25.093c-3.751,0-6.792-3.041-6.792-6.792c0-3.751,3.041-6.792,6.792-6.792h25.093
L376.322,461.82L376.322,461.82z" style="fill:#28549C;" />
  </g>
</svg>
```

Tool



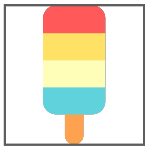
What geometric shape is the eraser of the pencil?

A: Rounded   B: Square
C: Circle   D: Triangle



```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512"
x="0px" y="0px" xml:space="preserve">
  <circle cx="256" cy="256" r="256" style="fill:#324A5E;" />
  <path d="M512,256c0-0.876-0.024-1.746-0.033-2.62L354.254,95.811L218.65,290.478l-60.787,105.427
l116.172,115.453C407.004,502.101,512,391.32,512,256z" style="fill:#2B3B4E;" />
  <path d="M338.965,402.244H173.037c-11.781,0-21.333-9.55-21.333-21.333V110.689
c0-11.781,9.55-21.333,21.333-21.333h165.928c11.781,0,21.333,9.55,21.333,21.333v270.224
C360.298,392.694,350.746,402.244,338.965,402.244z" style="fill:#F9B54C;" />
  <path d="M338.965,89.355h-82.963v312.889h82.963c11.781,0,21.333-9.55,21.333-21.333V110.689
C360.298,98.907,350.746,89.355,338.965,89.355z" style="fill:#F4A200;" />
  <rect height="69.525" style="fill:#90DFAA;" width="170.667" x="170.667" y="108.318" />
  <rect height="69.525" style="fill:#70C187;" width="85.333" x="256" y="108.318" />
  <g>
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="170.667" y="206.3" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="170.667" y="252.104" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="170.667" y="296.546" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="170.667" y="341.006" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="216.488" y="206.3" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="216.488" y="252.104" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="216.488" y="296.546" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="216.488" y="341.006" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="262.327" y="206.3" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="262.327" y="252.104" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="262.327" y="296.546" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="262.327" y="341.006" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="308.148" y="206.3" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="308.148" y="252.104" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="308.148" y="296.546" />
    <rect height="29.636" style="fill:#E6F3FF;" width="33.185" x="308.148" y="341.006" />
  </g>
</svg>
```

Tool



How many rows of buttons are there on the object in the image?

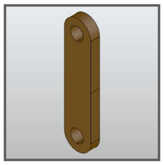
A: 2   B: 3
C: 4   D: 5

Figure 21: SVG examples in our SGP-Bench.

D.2 CAD Data

```
SOL;
Arc:(153,128,128,1);
Line:(153,169);
Line:(153,211);
Arc:(128,211,128,1);
Line:(128,169);
Line:(128,128);
SOL;
Circle:(140,128,6);
SOL;
Circle:(140,211,6);
Ext:(192,64,192,106,128,54,170,143,128,
Newbody,One-sided);
EOS
```

1 Operation

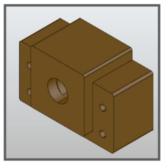


How many visible cylindrical sections does the CAD object have?

A: One B: Two
C: Three D: Four

```
SOL;
Arc:(130,126,64,1);Line:(221,126);Arc:(223,128,64,1);
Line:(223,176);Arc:(221,178,64,1);Line:(130,178);
Arc:(128,176,64,1);Line:(128,128);
SOL;
Circle:(139,138,3);
SOL;
Circle:(139,161,3);
SOL;
Circle:(176,161,10);
SOL;
Circle:(212,138,3);
SOL;
Circle:(212,161,3);
Ext:(192,64,192,34,128,62,189,207,128,Newbody,One-sided);
SOL;
Arc:(131,125,64,1);Line:(220,125);Arc:(223,128,64,1);
Line:(223,217);Arc:(220,220,64,1);Line:(131,220);
Arc:(128,217,64,1);Line:(128,128);
SOL;
Circle:(176,173,18);
Ext:(192,64,192,74,128,78,107,224,128,Join,One-sided);EOS
```

2 Operations

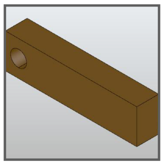


How many visible screws or screw holes are there on the CAD object?

A: Two B: Three
C: Four D: Five

```
SOL;
Line:(223,128);
Line:(223,153);
Line:(128,153);
Line:(128,128);
Ext:(192,64,192,32,128,83,174,157,128,
Newbody,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,64,192,39,99,106,21,12,128,
Cut,One-sided);
EOS
```

2 Operations

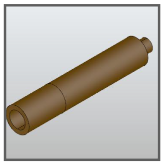


What type of feature is present on the side of the CAD object?

A: indentation B: protrusion
C: hole D: bump

```
SOL;
Arc:(131,125,64,1);Line:(211,125);Arc:(214,122,64,0);Line:(214,42);
Arc:(216,40,64,1);Line:(220,40);Arc:(223,42,64,1);Line:(223,132);
Arc:(220,135,64,1);Line:(131,135);Arc:(128,132,64,1);Line:(128,128);
Ext:(192,64,192,109,128,146,19,223,128,Newbody,One-sided);
SOL;
Arc:(131,125,64,1);Line:(211,125);Arc:(214,122,64,0);Line:(214,42);
Arc:(216,40,64,1);Line:(220,40);Arc:(223,42,64,1);Line:(223,132);
Arc:(220,135,64,1);Line:(131,135);Arc:(128,132,64,1);Line:(128,128);
Ext:(192,64,192,109,33,146,19,129,128,Newbody,One-sided);
SOL;
Arc:(129,127,64,1);Line:(222,127);Arc:(223,128,64,1);Line:(223,182);
Arc:(222,184,64,1);Line:(129,184);Arc:(128,182,64,1);Line:(128,128);
SOL;
Circle:(140,143,4);
SOL;
Circle:(140,167,4);
SOL;
Arc:(153,174,64,1);Line:(193,174);Arc:(194,173,64,0);Line:(194,133);
Arc:(196,131,64,1);Line:(198,131);Arc:(199,133,64,1);Line:(199,177);
Arc:(198,179,64,1);Line:(153,179);Arc:(152,177,64,1);Line:(152,176);
SOL;
Circle:(211,143,4);
SOL;
Circle:(211,167,4);
Ext:(192,64,192,99,33,127,38,129,128,Join,One-sided);
EOS
```

3 Operations

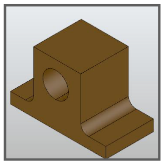


How many holes are visible on the flange of the CAD object?

A: Two B: Three
C: Four D: Five

```
SOL;
Line:(223,128);Line:(223,139);Line:(206,139);
Arc:(198,146,64,0);Line:(198,189);Line:(151,189);
Line:(151,146);Arc:(143,139,64,0);Line:(128,139);
Line:(128,128);
Ext:(192,64,192,69,128,128,148,202,128,
Newbody,One-sided);
SOL;
Circle:(176,128,47);
Ext:(192,64,192,119,54,189,40,4,128,
Cut,One-sided);
SOL;
Circle:(176,128,48);
Ext:(128,128,128,71,92,144,17,4,128,
Cut,One-sided);
SOL;
Circle:(176,128,48);
Ext:(128,128,128,198,90,144,16,4,128,
Cut,One-sided);
EOS
```

4 Operations

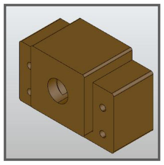


What feature is present at the base of the CAD object?

A: gears B: surface
C: fins D: recess

```
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,24);
Ext:(192,64,192,116,128,128,24,43,128,
Newbody,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,64,192,116,128,128,24,166,128,
Newbody,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,64,192,120,90,128,16,99,128,
Cut,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,192,192,134,128,128,12,224,128,
Join,One-sided);
EOS
```

4 Operations

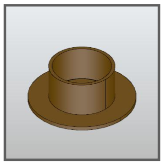


What shape is the primary body of the CAD object?

A: Square B: Triangle
C: Cylinder D: Sphere

```
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,29);
Ext:(128,128,128,32,128,128,192,135,128,Newbody,One-sided);
SOL;
Circle:(176,128,47);
SOL;
Circle:(176,128,5);
Ext:(128,128,128,75,128,128,106,135,128,Join,One-sided);
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,43);
Ext:(128,128,128,69,128,128,118,135,128,Join,One-sided);
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,43);
Ext:(128,128,128,69,128,128,118,200,128,Join,One-sided);
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,29);
Ext:(128,128,128,32,128,128,192,135,128,Join,One-sided);
EOS
```

5 Operations

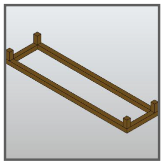


What is the shape of the base of the CAD object?

A: Triangle B: Square
C: Rectangle D: Circle

```
SOL;
Line:(223,128);Line:(223,153);Line:(128,153);
Line:(128,128);SOL;Line:(220,131);Line:(220,150);
Line:(131,150);Line:(131,131);
Ext:(128,128,128,128,128,128,96,131,128,Newbody,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,128,150,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,128,128,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,221,150,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,221,128,131,3,134,128,Join,One-sided);
EOS
```

5 Operations



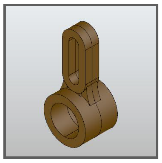
How many vertical supports does the CAD object have?

A: Two B: Four
C: Six D: Eight

Figure 22: CAD examples (3D) in our SGP-Bench.

```
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,32);
Ext:(192,64,192,101,128,128,54,149,128,Newbody,Symmetric);
SOL;
Arc:(181,128,128,1);Line:(181,197);Arc:(128,197,128,1);
Line:(128,128);
SOL;
Arc:(166,128,128,1);Line:(166,197);Arc:(143,197,128,1);
Line:(143,128);
Ext:(192,64,192,113,128,170,54,137,128,Join,Symmetric);
SOL;
Arc:(223,146,15,0);Arc:(135,177,27,0);Arc:(128,128,85,0);
Ext:(192,64,192,118,128,153,10,137,128,Join,Symmetric);
SOL;
Arc:(223,110,15,0);Arc:(216,158,85,0);Arc:(128,128,27,0);
Ext:(192,64,192,128,128,155,10,137,128,Join,Symmetric);
SOL;
Arc:(174,165,24,0);Arc:(177,184,85,1);
Arc:(156,223,33,0);Arc:(128,128,26,0);
Ext:(192,64,192,106,128,143,25,135,128,Join,Symmetric);
SOL;
Arc:(133,95,85,1);Arc:(211,33,24,0);Arc:(164,193,26,0);
Arc:(128,128,33,0);
Ext:(192,64,192,137,128,158,15,135,128,Join,Symmetric);EOS
```

6 Operations

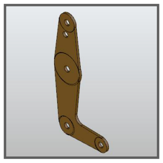


What type of geometric feature is the central hole of the CAD object?

A: Square B: Hexagonal
C: Circular D: Triangular

```
SOL;
Arc:(130,126,64,1);Line:(221,126);Arc:(223,128,64,1);
Line:(223,176);Arc:(221,178,64,1);Line:(130,178);
Arc:(128,176,64,1);Line:(128,128);
SOL;
Circle:(139,138,3);
SOL;
Circle:(139,161,3);
SOL;
Circle:(176,161,10);
SOL;
Circle:(212,138,3);
SOL;
Circle:(212,161,3);
Ext:(192,64,192,34,128,62,189,207,128,Newbody,One-sided);
SOL;
Arc:(131,125,64,1);Line:(220,125);Arc:(223,128,64,1);
Line:(223,217);Arc:(220,220,64,1);Line:(131,220);
Arc:(128,217,64,1);Line:(128,128);
SOL;
Circle:(176,173,18);
Ext:(192,64,192,74,128,78,107,224,128,Join,One-sided);EOS
```

6 Operations

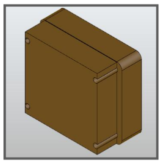


What is the primary shape of the CAD object?

A: Circular B: Triangular
C: Rectangular D: L-shaped

```
SOL;
Arc:(134,122,64,1);Line:(217,122);Arc:(223,128,64,1);
Line:(223,198);Arc:(217,204,64,1);Line:(134,204);
Arc:(128,198,64,1);Line:(128,128);
Ext:(192,64,192,32,128,57,192,82,128,Newbody,One-sided);
SOL;
Line:(223,128);Line:(223,220);Line:(128,220);
Line:(128,128);
Ext:(192,64,192,43,128,46,169,174,128,Join,One-sided);
SOL;
Circle:(176,128,47);
Ext:(192,192,192,214,174,194,11,14,128,Cut,One-sided);
SOL;
Circle:(176,128,47);
Ext:(192,192,192,54,174,194,11,14,128,Cut,One-sided);
SOL;
Circle:(176,128,47);
Ext:(192,192,192,214,174,62,11,14,128,Cut,One-sided);
SOL;
Circle:(176,128,47);
Ext:(192,192,192,54,174,62,11,14,128,Cut,One-sided);
SOL;
Line:(223,128);Line:(223,216);Line:(128,216);Line:(128,128);
Ext:(192,192,192,190,174,71,123,91,128,Cut,One-sided);
EOS
```

7 Operations

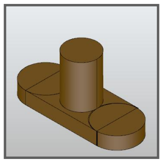


What shape primarily makes up the body of the CAD object?

A: Cylinder B: Sphere
C: Cube D: Cone

```
SOL;
Line:(128,223);Arc:(128,128,128,1);
Ext:(128,128,128,122,83,128,45,142,128,Newbody,One-sided);
SOL;
Arc:(128,223,128,1);Line:(128,128);
Ext:(128,128,128,122,83,128,45,142,128,Join,One-sided);
SOL;
Line:(223,128);Arc:(223,182,128,0);
Line:(128,182);Arc:(128,128,128,0);
Ext:(128,128,128,122,83,128,80,142,128,Join,One-sided);
SOL;
Arc:(128,223,128,1);Line:(128,128);
Ext:(128,128,128,201,83,128,45,142,128,Join,One-sided);
SOL;
Line:(128,223);Arc:(128,128,128,1);
Ext:(128,128,128,201,83,128,45,142,128,Join,One-sided);
SOL;
Circle:(176,128,48);
Ext:(128,128,128,145,105,142,34,176,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);Line:(128,128);
Ext:(255,128,255,149,118,128,26,118,128,Cut,One-sided);
EOS
```

7 Operations

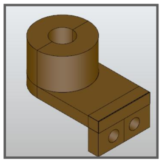


What shape is the primary feature of the CAD object?

A: Square B: Triangle
C: Cylinder D: Cone

```
SOL;
Line:(223,128);Line:(223,207);
Line:(128,207);Arc:(128,128,128,0);
Ext:(128,128,128,76,84,128,105,145,128,Newbody,One-sided);
SOL;Line:(128,156);Arc:(128,194,128,0);
Line:(128,223);Arc:(128,128,128,1);
Ext:(128,128,128,76,84,128,87,145,128,Join,One-sided);
SOL;Arc:(128,223,128,1);Line:(128,194);
Arc:(128,156,128,0);Line:(128,128);
Ext:(128,128,128,76,84,128,87,145,128,Join,One-sided);
SOL;Line:(128,156);Arc:(128,194,128,0);
Line:(128,223);Arc:(128,128,128,1);
Ext:(128,128,128,76,84,128,87,198,128,Join,One-sided);
SOL;Arc:(128,223,128,1);Line:(128,194);
Arc:(128,156,128,0);Line:(128,128);
Ext:(128,128,128,76,84,128,87,198,128,Join,One-sided);
SOL;Line:(176,128);Line:(223,128);Line:(223,147);
Line:(128,147);Line:(128,128);
Ext:(192,128,192,180,84,128,87,111,128,Join,One-sided);
SOL;Line:(223,128);Line:(223,204);
Line:(128,204);Line:(128,128);
SOL;Circle:(185,166,19);
Ext:(192,128,192,180,84,93,44,111,128,Join,One-sided);
SOL;Line:(223,128);Line:(223,204);
Line:(128,204);Line:(128,128);
SOL;Circle:(166,166,19);
Ext:(192,128,192,180,128,93,44,111,128,Join,One-sided);
EOS
```

8 Operations

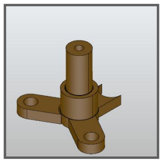


What is the shape of the main vertical feature in the CAD object?

A: Cube B: Sphere
C: Cylinder D: Cone

```
SOL;
Circle:(176,128,48);
SOL;
Circle:(176,128,24);
Ext:(192,64,192,116,128,128,24,43,128,
Newbody,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,64,192,116,128,128,24,166,128,
Newbody,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,64,192,120,90,128,16,99,128,
Cut,One-sided);
SOL;
Circle:(176,128,48);
Ext:(192,192,192,134,128,128,12,224,128,
Join,One-sided);
EOS
```

8 Operations

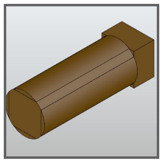


How many arms does the CAD object have extending from its central body?

A: One B: Two
C: Three D: Four

```
SOL;
Line:(223,128);Line:(223,223);
Line:(128,223);Line:(128,128);
Ext:(192,64,192,46,128,194,27,145,128,Newbody,One-sided);
SOL;Line:(223,128);Arc:(128,128,50,1);
Ext:(192,64,192,50,111,221,19,206,128,Join,One-sided);
SOL;Line:(128,223);Arc:(128,128,50,1);
Ext:(192,64,192,46,111,198,19,206,128,Join,One-sided);
SOL;Arc:(223,128,50,1);Line:(128,128);
Ext:(192,64,192,50,111,194,19,206,128,Join,One-sided);
SOL;Arc:(128,223,50,1);Line:(128,128);
Ext:(192,64,192,73,111,198,19,206,128,Join,One-sided);
SOL;Arc:(142,114,14,1);Line:(209,114);Arc:(223,128,14,1);
Line:(223,195);Arc:(209,209,14,1);Line:(142,209);
Arc:(128,195,14,1);Line:(128,128);
Ext:(192,64,192,46,111,198,27,206,128,Join,One-sided);
SOL;Circle:(176,128,48);
Ext:(128,128,128,54,89,221,11,111,183,Cut,Two-sided);
SOL;Circle:(176,128,48);
Ext:(128,128,128,54,67,221,11,111,183,Cut,Two-sided);
SOL;Circle:(176,128,48);
Ext:(128,128,128,54,45,221,11,111,183,Cut,Two-sided);EOS
```

9 Operations

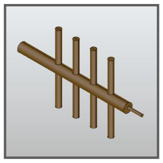


What shape is the main body of the CAD object?

A: Square B: Cylinder
C: Sphere D: Cone

```
SOL;
Line:(223,128);Line:(223,153);Line:(128,153);
Line:(128,128);SOL;Line:(220,131);Line:(220,150);
Line:(131,150);Line:(131,131);
Ext:(128,128,128,128,128,128,96,131,128,Newbody,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,128,150,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,128,128,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,221,150,131,3,134,128,Join,One-sided);
SOL;
Line:(223,128);Line:(223,223);Line:(128,223);
Line:(128,128);
Ext:(128,128,128,221,128,131,3,134,128,Join,One-sided);
EOS
```

10 Operations

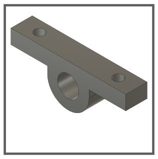


How many cylindrical rods are perpendicular to the main horizontal cylinder in the CAD image?

A: 3 B: 4
C: 5 D: 6

Figure 23: CAD examples (3D) in our SGP-Bench.

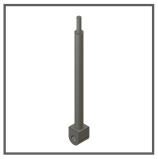
```
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Arc([0.46, 0.48, 0.00],
          [-0.46, 0.48, 0.00],
          [0.00, 0.48, 0.00],
          0.46,
          [0.00, 0.00, 1.00, 1.00],
          0.000,
          3.142,
          [1.00, -0.00, 0.00, 1.00]),
        Line([-0.46, 0.48, 0.00], [-0.46, 0.32, 0.00]),
        Line([-0.46, 0.32, 0.00], [-1.27, 0.32, 0.00]),
        Line([-1.27, 0.32, 0.00], [-1.27, 0.00, 0.00]),
        Line([-1.27, 0.00, 0.00], [1.27, 0.00, 0.00]),
        Line([1.27, 0.00, 0.00], [1.27, 0.32, 0.00]),
        Line([1.27, 0.32, 0.00], [0.46, 0.32, 0.00]),
        Line([0.46, 0.48, 0.00], [0.46, 0.32, 0.00]),
        is_outer=True
      ],
      LOOP[
        Circle([0.00, 0.48, 0.00], 0.24, [0.00, 0.00, 1.00, 1.00]),
        is_outer=False
      ]
    ],
    transform=Transform([0.00, 0.00, 0.00],
      [1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=ConstructionPlane(
      XZ,
      Plane([0.00, 0.00, 0.00],
        [0.00, 1.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00],
        [1.00, 0.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=NewBodyFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.475, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Circle([-0.87, 0.24, 0.00], 0.11, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform(
      [0.00, 0.00, -0.32],
      [-1.00, 0.00, 0.00, 1.00],
      [0.00, 1.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00]),
    reference_plane=BRepFace(
      [0.86, 0.24, -0.32],
      Plane(
        [1.27, 0.00, -0.32],
        [0.00, 0.00, -1.00, 1.00],
        [-1.00, 0.00, 0.00, 1.00],
        [0.00, 1.00, 0.00, 1.00]))));

SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Circle([0.87, 0.24, 0.00], 0.11, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform(
      [0.00, 0.00, -0.32],
      [-1.00, 0.00, 0.00, 1.00],
      [0.00, 1.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00]),
    reference_plane=BRepFace(
      [0.86, 0.24, -0.32],
      Plane(
        [1.27, 0.00, -0.32],
        [0.00, 0.00, -1.00, 1.00],
        [-1.00, 0.00, 0.00, 1.00],
        [0.00, 1.00, 0.00, 1.00]))));

ExtrudeFeature(
  operation=CutFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=-0.889, taper_angle=0.000));
```



How many holes are visible on the CAD object?

A: One B: Two
C: Three D: Four

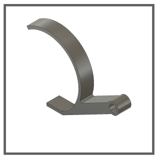
```
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Circle([0.00, 0.24, 0.00], 0.16, [0.00, 0.00, 1.00, 1.00]),
        is_outer=False],
      LOOP[
        Line([0.32, 0.24, 0.00], [0.32, -0.24, 0.00]),
        Arc(
          [0.32, 0.24, 0.00],
          [-0.32, 0.24, 0.00],
          [0.00, 0.24, 0.00],
          0.32,
          [0.00, 0.00, 1.00, 1.00],
          0.000,
          3.168,
          [1.00, -0.01, 0.00, 1.00]),
        Line([-0.32, -0.24, 0.00], [-0.32, 0.24, 0.00]),
        Line([0.32, -0.24, 0.00], [-0.32, -0.24, 0.00]),
        is_outer=True]],
    transform=Transform(
      [0.00, 0.00, 0.00],
      [1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=ConstructionPlane(
      XZ,
      Plane(
        [0.00, 0.00, 0.00],
        [0.00, 1.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00],
        [1.00, 0.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=NewBodyFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.635, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Circle([0.00, 0.32, 0.00], 0.20, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform(
      [0.00, 0.00, 0.24],
      [1.00, 0.00, -0.00, 1.00],
      [0.00, 1.00, 0.00, 1.00],
      [0.00, 0.00, 1.00, 1.00]),
    reference_plane=BRepFace(
      [-0.00, 0.32, 0.24],
      Plane(
        [-0.32, 0.00, 0.24],
        [0.00, -0.00, 1.00, 1.00],
        [1.00, 0.00, -0.00, 1.00],
        [0.00, 1.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=JoinFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=5.080, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Circle([-0.00, 0.32, 0.00], 0.11, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform(
      [0.00, 0.00, 5.32],
      [1.00, 0.00, -0.00, 1.00],
      [0.00, 1.00, 0.00, 1.00],
      [0.00, 0.00, 1.00, 1.00]),
    reference_plane=BRepFace(
      [0.00, 0.32, 5.32],
      Plane(
        [0.00, 0.32, 5.32],
        [0.00, -0.00, 1.00, 1.00],
        [1.00, 0.00, -0.00, 1.00],
        [0.00, 1.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=JoinFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.792, taper_angle=0.000));
```



How many holes are visible on the CAD object?

A: One B: Two
C: Three D: Four

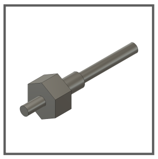
```
SKETCH(
  PROFILE(
    loops=[
      LOOP[
        Line([4.90, -0.22, 0.00], [2.81, 0.87, 0.00]),
        Arc([4.90, -0.22, 0.00],
          [5.18, 0.42, 0.00],
          [5.30, -0.01, 0.00],
          0.45,
          [0.00, 0.00, 1.00, 1.00],
          0.000,
          4.494,
          [-0.88, -0.47, 0.00, 1.00]),
        Line([1.28, 2.46, 0.00], [5.18, 0.42, 0.00]),
        Line([1.28, 2.46, 0.00], [0.00, 2.46, 0.00]),
        Line([0.00, 2.46, 0.00], [0.00, 2.24, 0.00]),
        Line([0.00, 2.24, 0.00], [1.23, 2.24, 0.00]),
        Arc([-1.46, -2.09, 0.00],
          [1.23, 2.24, 0.00],
          [0.00, -0.00, 0.00],
          2.55,
          [0.00, 0.00, 1.00, 1.00],
          0.000,
          3.251,
          [-0.57, -0.82, 0.00, 1.00]),
        Line([-1.59, -2.27, 0.00], [-1.46, -2.09, 0.00]),
        Arc([-1.59, -2.27, 0.00],
          [2.67, 0.75, 0.00],
          [-0.00, 0.00, 0.00],
          2.77,
          [0.00, 0.00, 1.00, 1.00],
          0.000,
          2.458,
          [-0.57, -0.82, 0.00, 1.00]),
        Arc([2.81, 0.87, 0.00],
          [2.67, 0.75, 0.00],
          [2.76, 0.78, 0.00],
          0.10,
          [-0.00, 0.00, 1.00, 1.00],
          0.000,
          2.327,
          [0.46, 0.89, 0.00, 1.00]),
        is_outer=True],
      LOOP[
        Circle([5.30, -0.01, 0.00], 0.30, [0.00, 0.00, 1.00, 1.00]),
        is_outer=False]],
    transform=Transform(
      [0.00, 0.00, 0.00],
      [1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=ConstructionPlane(
      XZ,
      Plane(
        [0.00, 0.00, 0.00],
        [0.00, 1.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00],
        [1.00, 0.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=NewBodyFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=1.000, taper_angle=0.000));
```



What is the orientation of the curved beam in relation to the base?

A: Perpendicular B: Parallel
C: Diagonal D: Inclined

```
SKETCH(
  PROFILE(
    loops=[LOOP[
        Line([-0.43, -0.25, 0.00], [0.00, -0.50, 0.00]),
        Line([0.00, -0.50, 0.00], [0.43, -0.25, 0.00]),
        Line([0.43, -0.25, 0.00], [0.43, 0.25, 0.00]),
        Line([0.43, 0.25, 0.00], [0.00, 0.50, 0.00]),
        Line([0.00, 0.50, 0.00], [-0.43, 0.25, 0.00]),
        Line([-0.43, 0.25, 0.00], [-0.43, -0.25, 0.00]),
        is_outer=True]],
    transform=Transform([0.00, 0.00, 0.00],
      [1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, -1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=ConstructionPlane(
      XZ,Plane([0.00, 0.00, 0.00],
        [0.00, 1.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00],
        [1.00, 0.00, 0.00, 1.00]))));
ExtrudeFeature(
  operation=NewBodyFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.509, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[LOOP[
        Circle([0.00, 0.00, 0.00], 0.21, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform([0.00, 0.51, 0.00],
      [-1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, 1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=BRepFace(
      [-0.00, 0.51, 0.00],
      Plane([-0.00, 0.51, 0.00],
        [0.00, 1.00, -0.00, 1.00],
        [-1.00, 0.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00]))));
ExtrudeFeature(
  operation=JoinFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.651, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[LOOP[
        Circle([0.00, 0.00, 0.00], 0.13, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform([0.00, 1.16, 0.00],
      [-1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, 1.00, 1.00],
      [0.00, 1.00, 0.00, 1.00]),
    reference_plane=BRepFace([0.00, 1.16, 0.00],
      Plane([0.00, 1.16, 0.00],
        [0.00, 1.00, -0.00, 1.00],
        [-1.00, 0.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00]))));
ExtrudeFeature(
  operation=JoinFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=1.759, taper_angle=0.000));
SKETCH(
  PROFILE(
    loops=[LOOP[
        Circle([0.00, 0.00, 0.00], 0.13, [0.00, 0.00, 1.00, 1.00]),
        is_outer=True]],
    transform=Transform([0.00, 0.00, 0.00],
      [1.00, 0.00, 0.00, 1.00],
      [0.00, 0.00, 1.00, 1.00],
      [0.00, -1.00, 0.00, 1.00]),
    reference_plane=BRepFace(
      [-0.00, 0.00, 0.00],
      Plane([-0.00, 0.00, 0.00],
        [0.00, 1.00, -0.00, 1.00],
        [-1.00, 0.00, 0.00, 1.00],
        [0.00, 0.00, 1.00, 1.00]))));
ExtrudeFeature(
  operation=JoinFeatureOperation,
  start_extent=ProfilePlaneStartDefinition,
  extent_type=OneSideFeatureExtentType,
  extent_one=DistanceExtentDefinition(distance=0.552, taper_angle=0.000));
```

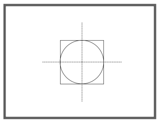


What is the orientation of the smaller cylinder relative to the larger cylinder in the CAD object?

A: Perpendicular B: Parallel
C: Acute angle D: Obtuse angle

Figure 24: CAD examples (3D$_{\text{complex}}$) in our SGP-Bench.

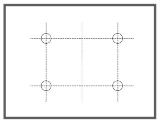
```
0:External;
1:Circle(xCenter=0.0000, yCenter=0.0150, xDir=1.0000, yDir=0.0000, radius=0.0150, clockwise=False,
isConstruction=False);
CoincidentConstraint(references=(1, 0));
DiameterConstraint(references=(1,), length=0.03 METER);
2:SN_Center;
SubnodeConstraint(references=(2, 1));
VerticalConstraint(references=(2, 0));
3:Line(dirX=-1.0000, dirY=-0.0000, pntX=0.0179, pntY=0.0300, startParam=0.0029, endParam=0.0179,
isConstruction=False);
TangentConstraint(references=(3, 1));
HorizontalConstraint(references=(3,));
4:SN_Start;
SubnodeConstraint(references=(4, 3));
5:SN_End;
SubnodeConstraint(references=(5, 3));
CoincidentConstraint(references=(5, 1));
6:Line(dirX=0.0000, dirY=-1.0000, pntX=0.0150, pntY=0.0221, startParam=-0.0079, endParam=0.0071,
isConstruction=False);
TangentConstraint(references=(6, 1));
7:SN_Start;
SubnodeConstraint(references=(7, 6));
CoincidentConstraint(references=(7, 4));
8:SN_End;
SubnodeConstraint(references=(8, 6));
CoincidentConstraint(references=(8, 1));
9:Line(dirX=0.0000, dirY=-1.0000, pntX=0.0150, pntY=0.0066, startParam=-0.0084, endParam=0.0066,
isConstruction=False);
VerticalConstraint(references=(9,));
10:SN_Start;
SubnodeConstraint(references=(10, 9));
CoincidentConstraint(references=(10, 8));
11:SN_End;
SubnodeConstraint(references=(11, 9));
VerticalConstraint(references=(11, 4));
12:Line(dirX=-1.0000, dirY=-0.0000, pntX=0.0094, pntY=0.0000, startParam=-0.0056, endParam=0.0094,
isConstruction=False);
TangentConstraint(references=(12, 1));
HorizontalConstraint(references=(12,));
13:SN_Start;
SubnodeConstraint(references=(13, 12));
CoincidentConstraint(references=(13, 11));
14:SN_End;
SubnodeConstraint(references=(14, 12));
CoincidentConstraint(references=(14, 1));
15:Line(dirX=-1.0000, dirY=-0.0000, pntX=-0.0076, pntY=-0.0000, startParam=-0.0076, endParam=0.0074,
isConstruction=False);
HorizontalConstraint(references=(15,));
16:SN_Start;
SubnodeConstraint(references=(16, 15));
CoincidentConstraint(references=(16, 14));
17:SN_End;
SubnodeConstraint(references=(17, 15));
18:Line(dirX=0.0000, dirY=1.0000, pntX=-0.0150, pntY=0.0066, startParam=-0.0066, endParam=-0.0066,
isConstruction=False);
TangentConstraint(references=(18, 1));
VerticalConstraint(references=(18,));
19:SN_Start;
SubnodeConstraint(references=(19, 18));
CoincidentConstraint(references=(19, 17));
20:SN_End;
SubnodeConstraint(references=(20, 18));
CoincidentConstraint(references=(20, 17));
21:Line(dirX=0.0000, dirY=1.0000, pntX=-0.0150, pntY=0.0236, startParam=-0.0236, endParam=0.0064,
isConstruction=False);
VerticalConstraint(references=(21,));
22:SN_Start;
SubnodeConstraint(references=(22, 21));
CoincidentConstraint(references=(22, 20));
23:SN_End;
SubnodeConstraint(references=(23, 21));
24:Line(dirX=1.0000, dirY=0.0000, pntX=-0.0098, pntY=0.0300, startParam=-0.0052, endParam=0.0098,
isConstruction=False);
HorizontalConstraint(references=(24,));
25:SN_Start;
SubnodeConstraint(references=(25, 24));
CoincidentConstraint(references=(25, 23));
26:SN_End;
SubnodeConstraint(references=(26, 24));
CoincidentConstraint(references=(26, 5));
27:Line(dirX=1.0000, dirY=0.0000, pntX=0.0036, pntY=0.0150, startParam=-0.0309, endParam=0.0236,
isConstruction=True);
HorizontalConstraint(references=(27,));
MidpointConstraint(references=(27, 2));
28:SN_Start;
SubnodeConstraint(references=(28, 27));
29:SN_End;
SubnodeConstraint(references=(29, 27));
30:Line(dirX=0.0000, dirY=-1.0000, pntX=0.0000, pntY=0.0108, startParam=-0.0315, endParam=0.0230,
isConstruction=True);
MidpointConstraint(references=(30, 2));
EqualConstraint(references=(30, 27));
31:SN_Start;
SubnodeConstraint(references=(31, 30));
32:SN_End;
SubnodeConstraint(references=(32, 30));
VerticalConstraint(references=(32, 14));
33:Stop;
```



What is the shape of the outermost figure in the image?

A: Circle B: Triangle
C: Square D: Rectangle

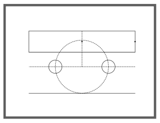
```
0:External;
1:Line(dirX=-1.0000, dirY=-0.0000, pntX=-0.0035, pntY=0.0319, startParam=-0.0295, endParam=0.0225,
isConstruction=True);
PerpendicularConstraint(references=(1, 0));
2:SN_Start;
SubnodeConstraint(references=(2, 1));
CoincidentConstraint(references=(2, 0));
3:SN_End;
SubnodeConstraint(references=(3, 1));
CoincidentConstraint(references=(3, 0));
4:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.0180, pntY=0.0199, startParam=-0.0199, endParam=0.0199,
isConstruction=True);
VerticalConstraint(references=(4,));
5:SN_Start;
SubnodeConstraint(references=(5, 4));
CoincidentConstraint(references=(5, 0));
6:SN_End;
SubnodeConstraint(references=(6, 4));
CoincidentConstraint(references=(6, 0));
7:Line(dirX=-0.6868, dirY=0.7268, pntX=-0.0181, pntY=0.0002, startParam=-0.0002, endParam=0.0002,
isConstruction=True);
8:SN_Start;
SubnodeConstraint(references=(8, 7));
CoincidentConstraint(references=(8, 6));
9:SN_End;
SubnodeConstraint(references=(9, 7));
10:Line(dirX=-1.0000, dirY=-0.0000, pntX=-0.0000, pntY=0.0080, startParam=-0.0260, endParam=0.0260,
isConstruction=True);
PerpendicularConstraint(references=(10, 0));
11:SN_Start;
SubnodeConstraint(references=(11, 10));
CoincidentConstraint(references=(11, 0));
12:SN_End;
SubnodeConstraint(references=(12, 10));
CoincidentConstraint(references=(12, 0));
13:Line(dirX=0.0000, dirY=-1.0000, pntX=0.0180, pntY=0.0199, startParam=-0.0199, endParam=0.0199,
isConstruction=True);
VerticalConstraint(references=(13,));
14:SN_Start;
SubnodeConstraint(references=(14, 13));
CoincidentConstraint(references=(14, 0));
15:SN_End;
SubnodeConstraint(references=(15, 13));
CoincidentConstraint(references=(15, 0));
16:Circle(xCenter=0.0180, yCenter=0.0319, xDir=1.0000, yDir=0.0000, radius=0.0025, clockwise=False,
isConstruction=False);
DiameterConstraint(references=(16,), length=0.005004 METER);
17:SN_Center;
SubnodeConstraint(references=(17, 16));
CoincidentConstraint(references=(17, 1));
CoincidentConstraint(references=(17, 13));
18:Circle(xCenter=0.0180, yCenter=0.0080, xDir=1.0000, yDir=0.0000, radius=0.0025, clockwise=False,
isConstruction=False);
EqualConstraint(references=(18, 16));
19:SN_Center;
SubnodeConstraint(references=(19, 18));
CoincidentConstraint(references=(19, 10));
CoincidentConstraint(references=(19, 13));
20:Circle(xCenter=-0.0180, yCenter=0.0080, xDir=1.0000, yDir=0.0000, radius=0.0025, clockwise=False,
isConstruction=False);
EqualConstraint(references=(20, 16));
21:SN_Center;
SubnodeConstraint(references=(21, 20));
CoincidentConstraint(references=(21, 10));
CoincidentConstraint(references=(21, 4));
22:Circle(xCenter=-0.0180, yCenter=0.0319, xDir=1.0000, yDir=0.0000, radius=0.0025, clockwise=False,
isConstruction=False);
EqualConstraint(references=(22, 16));
23:SN_Center;
SubnodeConstraint(references=(23, 22));
CoincidentConstraint(references=(23, 4));
CoincidentConstraint(references=(23, 1));
24:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.0000, pntY=0.0199, startParam=-0.0199, endParam=0.0199,
isConstruction=True);
VerticalConstraint(references=(24,));
25:SN_Start;
SubnodeConstraint(references=(25, 24));
CoincidentConstraint(references=(25, 0));
26:SN_End;
SubnodeConstraint(references=(26, 24));
HorizontalConstraint(references=(26, 0));
27:Line(dirX=0.0000, dirY=1.0000, pntX=-0.0000, pntY=0.0259, startParam=-0.0060, endParam=0.0060,
isConstruction=True);
28:SN_Start;
SubnodeConstraint(references=(28, 27));
MidpointConstraint(references=(28, 24));
29:SN_End;
SubnodeConstraint(references=(29, 27));
MidpointConstraint(references=(29, 1));
30:Stop;
```



How many circles are shown in the image?

A: Three B: Four
C: Five D: Six

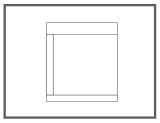
```
0:External;
1:Line(dirX=-1.0000, dirY=-0.0000, pntX=0.3174, pntY=-0.2227, startParam=-0.2543, endParam=0.2543,
isConstruction=False);
HorizontalConstraint(references=(1,));
2:SN_Start;
SubnodeConstraint(references=(2, 1));
CoincidentConstraint(references=(2, 0));
3:SN_End;
SubnodeConstraint(references=(3, 1));
CoincidentConstraint(references=(3, 0));
4:Line(dirX=-1.0000, dirY=-0.0000, pntX=0.3174, pntY=-0.2227, startParam=-0.2543, endParam=0.2543,
isConstruction=False);
5:SN_Start;
SubnodeConstraint(references=(5, 4));
CoincidentConstraint(references=(5, 2));
6:SN_End;
SubnodeConstraint(references=(6, 4));
7:Line(dirX=-1.0000, dirY=0.0000, pntX=0.3174, pntY=-0.1211, startParam=-0.2543, endParam=0.2543,
isConstruction=False);
ParallelConstraint(references=(7, 4));
HorizontalConstraint(references=(7,));
8:SN_Start;
SubnodeConstraint(references=(8, 7));
9:SN_End;
SubnodeConstraint(references=(9, 7));
CoincidentConstraint(references=(9, 0));
10:Line(dirX=0.0000, dirY=1.0000, pntX=0.5717, pntY=-0.1846, startParam=-0.0381, endParam=0.0635,
isConstruction=False);
PerpendicularConstraint(references=(10, 7));
11:SN_Start;
SubnodeConstraint(references=(11, 10));
CoincidentConstraint(references=(11, 5));
12:SN_End;
SubnodeConstraint(references=(12, 10));
CoincidentConstraint(references=(12, 8));
13:Line(dirX=0.0000, dirY=1.0000, pntX=0.0631, pntY=-0.1846, startParam=-0.0381, endParam=0.0635,
isConstruction=False);
ParallelConstraint(references=(13, 10));
14:SN_Start;
SubnodeConstraint(references=(14, 13));
CoincidentConstraint(references=(14, 6));
15:SN_End;
SubnodeConstraint(references=(15, 13));
CoincidentConstraint(references=(15, 9));
16:Line(dirX=-1.0000, dirY=0.0000, pntX=0.3174, pntY=-0.4196, startParam=-0.2543, endParam=0.2543,
isConstruction=False);
ParallelConstraint(references=(16, 0));
17:SN_Start;
SubnodeConstraint(references=(17, 16));
18:SN_End;
SubnodeConstraint(references=(18, 16));
VerticalConstraint(references=(18, 0));
19:Point(x=0.5717, y=-0.1719, isConstruction=False);
MidpointConstraint(references=(19, 10));
VerticalConstraint(references=(19, 17));
20:Line(dirX=-1.0000, dirY=-0.0000, pntX=0.3174, pntY=-0.2926, startParam=-0.2543, endParam=0.2543,
isConstruction=True);
HorizontalConstraint(references=(20,));
21:SN_Start;
SubnodeConstraint(references=(21, 20));
CoincidentConstraint(references=(21, 0));
22:SN_End;
SubnodeConstraint(references=(22, 20));
CoincidentConstraint(references=(22, 0));
23:Circle(xCenter=0.3174, yCenter=-0.2926, xDir=1.0000, yDir=0.0000, radius=0.1270, clockwise=False,
isConstruction=True);
DiameterConstraint(references=(23,), length=0.254 METER);
24:SN_Center;
SubnodeConstraint(references=(24, 23));
MidpointConstraint(references=(24, 20));
25:Circle(xCenter=0.4444, yCenter=-0.2926, xDir=1.0000, yDir=0.0000, radius=0.0318, clockwise=False,
isConstruction=False);
DiameterConstraint(references=(25,), length=0.0635 METER);
26:SN_Center;
SubnodeConstraint(references=(26, 25));
CoincidentConstraint(references=(26, 20));
CoincidentConstraint(references=(26, 23));
27:Circle(xCenter=0.1904, yCenter=-0.2926, xDir=1.0000, yDir=0.0000, radius=0.0318, clockwise=False,
isConstruction=False);
DiameterConstraint(references=(27,), length=0.0635 METER);
28:SN_Center;
SubnodeConstraint(references=(28, 27));
CoincidentConstraint(references=(28, 20));
CoincidentConstraint(references=(28, 23));
29:Line(dirX=-0.0000, dirY=1.0000, pntX=0.3174, pntY=-0.2069, startParam=-0.0857, endParam=0.0857,
isConstruction=True);
30:SN_Start;
SubnodeConstraint(references=(30, 29));
CoincidentConstraint(references=(30, 24));
31:SN_End;
SubnodeConstraint(references=(31, 29));
MidpointConstraint(references=(31, 7));
32:Point(x=0.3174, y=-0.1719, isConstruction=False);
CoincidentConstraint(references=(32, 29));
33:Stop;
```



Are the smaller circles inside or outside of the larger circle?

A: Inside B: Outside
C: On border D: Not determinable

```
0:External;
1:Line(dirX=1.0000, dirY=0.0000, pntX=-0.3085, pntY=0.3702, startParam=-0.0020, endParam=0.0190, isConstruction=False);
2:SN_Start;
SubnodeConstraint(references=(2, 1));
CoincidentConstraint(references=(2, 0));
3:SN_End;
SubnodeConstraint(references=(3, 1));
4:Line(dirX=1.0000, dirY=0.0000, pntX=-0.3057, pntY=0.3487, startParam=-0.0048, endParam=0.0162, isConstruction=False);
ParallelConstraint(references=(4, 1));
HorizontalConstraint(references=(4,));
5:SN_Start;
SubnodeConstraint(references=(5, 4));
6:SN_End;
SubnodeConstraint(references=(6, 4));
7:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.3105, pntY=0.3633, startParam=-0.0069, endParam=0.0146, isConstruction=False);
PerpendicularConstraint(references=(7, 4));
8:SN_Start;
SubnodeConstraint(references=(8, 7));
CoincidentConstraint(references=(8, 2));
9:SN_End;
SubnodeConstraint(references=(9, 7));
CoincidentConstraint(references=(9, 5));
10:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.2895, pntY=0.3633, startParam=-0.0069, endParam=0.0146, isConstruction=False);
ParallelConstraint(references=(10, 7));
11:SN_Start;
SubnodeConstraint(references=(11, 10));
CoincidentConstraint(references=(11, 3));
12:SN_End;
SubnodeConstraint(references=(12, 10));
CoincidentConstraint(references=(12, 6));
13:Line(dirX=-1.0000, dirY=0.0000, pntX=-0.3000, pntY=0.3487, startParam=-0.0105, endParam=0.0105, isConstruction=False);
14:SN_Start;
SubnodeConstraint(references=(14, 13));
CoincidentConstraint(references=(14, 6));
15:SN_End;
SubnodeConstraint(references=(15, 13));
16:Line(dirX=-1.0000, dirY=-0.0000, pntX=-0.3000, pntY=0.3467, startParam=-0.0105, endParam=0.0105, isConstruction=False);
ParallelConstraint(references=(16, 13));
HorizontalConstraint(references=(16,));
17:SN_Start;
SubnodeConstraint(references=(17, 16));
18:SN_End;
SubnodeConstraint(references=(18, 16));
VerticalConstraint(references=(18, 5));
19:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.2895, pntY=0.3477, startParam=-0.0010, endParam=0.0010, isConstruction=False);
PerpendicularConstraint(references=(19, 16));
20:SN_Start;
SubnodeConstraint(references=(20, 19));
CoincidentConstraint(references=(20, 14));
21:SN_End;
SubnodeConstraint(references=(21, 19));
CoincidentConstraint(references=(21, 17));
22:Line(dirX=0.0000, dirY=-1.0000, pntX=-0.3105, pntY=0.3477, startParam=-0.0010, endParam=0.0010, isConstruction=False);
ParallelConstraint(references=(22, 19));
23:SN_Start;
SubnodeConstraint(references=(23, 22));
CoincidentConstraint(references=(23, 15));
24:SN_End;
SubnodeConstraint(references=(24, 22));
CoincidentConstraint(references=(24, 18));
25:Line(dirX=1.0000, dirY=0.0000, pntX=-0.3000, pntY=0.3667, startParam=-0.0105, endParam=0.0105, isConstruction=False);
PerpendicularConstraint(references=(25, 10));
26:SN_Start;
SubnodeConstraint(references=(26, 25));
CoincidentConstraint(references=(26, 7));
27:SN_End;
SubnodeConstraint(references=(27, 25));
CoincidentConstraint(references=(27, 10));
28:Line(dirX=1.0000, dirY=0.0000, pntX=-0.3155, pntY=0.3487, startParam=0.0050, endParam=0.0070, isConstruction=False);
29:SN_Start;
SubnodeConstraint(references=(29, 28));
CoincidentConstraint(references=(29, 5));
30:SN_End;
SubnodeConstraint(references=(30, 28));
31:Line(dirX=1.0000, dirY=0.0000, pntX=-0.3155, pntY=0.3667, startParam=0.0050, endParam=0.0070, isConstruction=False);
ParallelConstraint(references=(31, 28));
HorizontalConstraint(references=(31,));
EqualConstraint(references=(31, 19));
32:SN_Start;
SubnodeConstraint(references=(32, 31));
33:SN_End;
SubnodeConstraint(references=(33, 31));
CoincidentConstraint(references=(33, 25));
34:Line(dirX=-0.0000, dirY=1.0000, pntX=-0.3105, pntY=0.3577, startParam=-0.0090, endParam=0.0090, isConstruction=False);
PerpendicularConstraint(references=(34, 31));
35:SN_Start;
SubnodeConstraint(references=(35, 34));
CoincidentConstraint(references=(35, 29));
36:SN_End;
SubnodeConstraint(references=(36, 34));
CoincidentConstraint(references=(36, 32));
37:Line(dirX=-0.0000, dirY=1.0000, pntX=-0.3085, pntY=0.3577, startParam=-0.0090, endParam=0.0090, isConstruction=False);
ParallelConstraint(references=(37, 34));
38:SN_Start;
SubnodeConstraint(references=(38, 37));
CoincidentConstraint(references=(38, 30));
39:SN_End;
SubnodeConstraint(references=(39, 37));
CoincidentConstraint(references=(39, 33));
40:Stop;
```

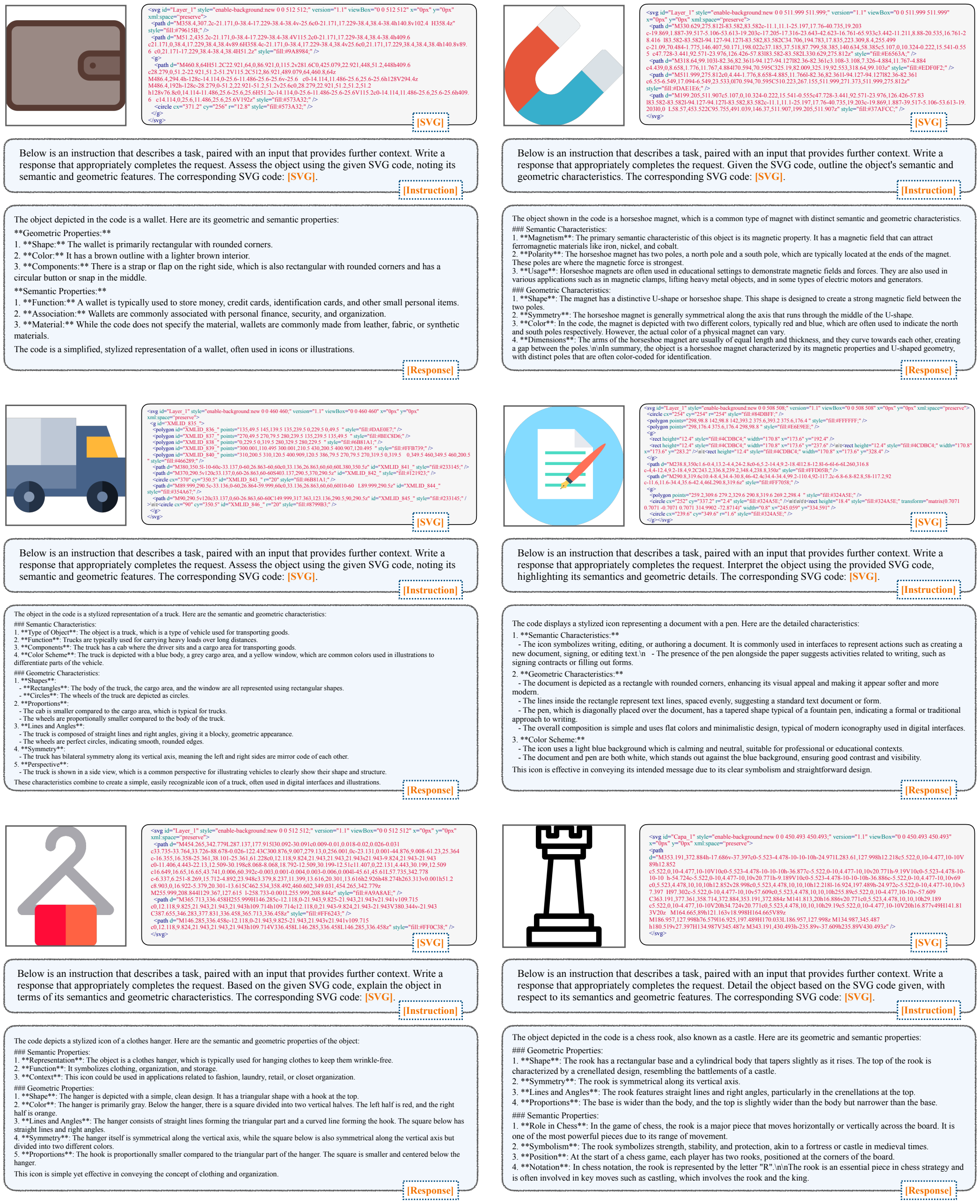


What type of angles are formed at the corners of the inner shape in the image?

A: Acute B: Right
C: Obtuse D: Straight

Figure 25: CAD examples (2D) in our SGP-Bench.

D.3 Symbolic Instruction-following Data (SVG)



```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <path d="M358.4,307.2c-21.171,0-38.4-17.229-38.4-38.4v-25.6c0-21.171,17.229-38.4,38.4-38.4h140.8v102.4  H358.4z"
style="fill:#79615B;" />
  <path d="M51.2,435.2c-21.171,0-38.4-17.229-38.4-38.4V115.2c0-21.171,17.229-38.4,38.4-38.4h409.6
c21.171,0,38.4,17.229,38.4,38.4v89.6H358.4c-21.171,0-38.4,17.229-38.4,38.4v25.6c0,21.171,17.229,38.4,38.4,38.4h140.8v89.
6  c0,21.171-17.229,38.4-38.4,38.4H51.2z" style="fill:#9A8984;" />
  <g>
    <path d="M460.8,64H51.2C22.921,64,0,86.921,0,115.2v281.6C0,425.079,22.921,448,51.2,448h409.6
c28.279,0,51.2-22.921,51.2-51.2V115.2C512,86.921,489.079,64,460.8,64z
M486.4,294.4h-128c-14.114,0-25.6-11.486-25.6-25.6v-25.6   c0-14.114,11.486-25.6,25.6-25.6h128V294.4z
M486.4,192h-128c-28.279,0-51.2,22.921-51.2,51.2v25.6c0,28.279,22.921,51.2,51.2,51.2
h128v76.8c0,14.114-11.486,25.6-25.6,25.6H51.2c-14.114,0-25.6-11.486-25.6-25.6V115.2c0-14.114,11.486-25.6,25.6-25.6h409.
6   c14.114,0,25.6,11.486,25.6,25.6V192z" style="fill:#573A32;" />
    <circle cx="371.2" cy="256" r="12.8" style="fill:#573A32;" />
  </g>
</svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Assess the object using the given SVG code, noting its semantic and geometric features. The corresponding SVG code: [SVG].

[Instruction]

The object depicted in the code is a wallet. Here are its geometric and semantic properties:

Geometric Properties:
1. **Shape:** The wallet is primarily rectangular with rounded corners.
2. **Color:** It has a brown outline with a lighter brown interior.
3. **Components:** There is a strap or flap on the right side, which is also rectangular with rounded corners and has a circular button or snap in the middle.

Semantic Properties:
1. **Function:** A wallet is typically used to store money, credit cards, identification cards, and other small personal items.
2. **Association:** Wallets are commonly associated with personal finance, security, and organization.
3. **Material:** While the code does not specify the material, wallets are commonly made from leather, fabric, or synthetic materials.

The code is a simplified, stylized representation of a wallet, often used in icons or illustrations.

[Response]

```
<svg id="Layer_1" style="enable-background:new 0 0 511.999 511.999;" version="1.1" viewBox="0 0 511.999 511.999"
x="0px" y="0px" xml:space="preserve">
  <path d="M330.629,275.812l-83.582,83.582c-11.1,11.1-25.197,17.76-40.735,19.203
c-19.869,1.887-39.517-5.106-53.613-19.203c-17.205-17.316-23.643-42.623-16.761-65.933c3.442-11.211,8.88-20.535,16.761-2
8.416  l83.582-83.582l-94.127-94.127l-83.582,83.582C34.706,194.783,17.835,223.309,8.4,255.499
c-21.09,70.484-1.775,146.407,50.171,198.022c37.185,37.518,87.799,58.385,140.634,58.385c5.107,0,10.324-0.222,15.541-0.55
5  c47.728-3.441,92.571-23.976,126.426-57.83l83.582-83.582L330.629,275.812z" style="fill:#E6563A;" />
  <path d="M318.64,99.103l-82.36,82.361l-94.127-94.127l82.36-82.361c3.108-3.108,7.326-4.884,11.767-4.884
c4.439,0,8.658,1.776,11.767,4.884l70.594,70.595C325.19,82.009,325.19,92.553,318.64,99.103z" style="fill:#EDF0F2;" />
  <path d="M511.999,275.812c0,4.44-1.776,8.658-4.885,11.766l-82.36,82.361l-94.127-94.127l82.36-82.361
c6.55-6.549,17.094-6.549,23.533,0l70.594,70.595C510.223,267.155,511.999,271.373,511.999,275.812z"
style="fill:#DAE1E6;" />
  <path d="M199.205,511.907c5.107,0,10.324-0.222,15.541-0.555c47.728-3.441,92.571-23.976,126.426-57.83
l83.582-83.582l-94.127-94.127l-83.582,83.582c-11.1,11.1-25.197,17.76-40.735,19.203c-19.869,1.887-39.517-5.106-53.613-19.
203l0,0  L58.57,453.522C95.755,491.039,146.37,511.907,199.205,511.907z" style="fill:#37AFCC;" />
</svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Given the SVG code, outline the object's semantic and geometric characteristics. The corresponding SVG code: [SVG].

[Instruction]

The object shown in the code is a horseshoe magnet, which is a common type of magnet with distinct semantic and geometric characteristics.
Semantic Characteristics:
1. **Magnetism**: The primary semantic characteristic of this object is its magnetic property. It has a magnetic field that can attract ferromagnetic materials like iron, nickel, and cobalt.
2. **Polarity**: The horseshoe magnet has two poles, a north pole and a south pole, which are typically located at the ends of the magnet. These poles are where the magnetic force is strongest.
3. **Usage**: Horseshoe magnets are often used in educational settings to demonstrate magnetic fields and forces. They are also used in various applications such as in magnetic clamps, lifting heavy metal objects, and in some types of electric motors and generators.

Geometric Characteristics:
1. **Shape**: The magnet has a distinctive U-shape or horseshoe shape. This shape is designed to create a strong magnetic field between the two poles.
2. **Symmetry**: The horseshoe magnet is generally symmetrical along the axis that runs through the middle of the U-shape.
3. **Color**: In the code, the magnet is depicted with two different colors, typically red and blue, which are often used to indicate the north and south poles respectively. However, the actual color of a physical magnet can vary.
4. **Dimensions**: The arms of the horseshoe magnet are usually of equal length and thickness, and they curve towards each other, creating a gap between the poles.\n\nIn summary, the object is a horseshoe magnet characterized by its magnetic properties and U-shaped geometry, with distinct poles that are often color-coded for identification.

[Response]

```
<svg id="Layer_1" style="enable-background:new 0 0 460 460;" version="1.1" viewBox="0 0 460 460" x="0px" y="0px"
xml:space="preserve">
  <g id="XMLID_835_">
    <polygon id="XMLID_836_" points="135,49.5 145,139.5 135,239.5 0,229.5 0,49.5  " style="fill:#DAE0E7;" />
    <polygon id="XMLID_837_" points="270,49.5 270,79.5 280,239.5 135,239.5 135,49.5  " style="fill:#BEC8D6;" />
    <polygon id="XMLID_838_" points="0,229.5 0,319.5 280,329.5 280,229.5  " style="fill:#6B81A1;" />
    <polygon id="XMLID_839_" points="300.001,110.495 300.001,210.5 430,200.5 400.907,120.495  " style="fill:#FFB739;" />
    <polygon id="XMLID_840_" points="310,200.5 310,120.5 400.909,120.5 386,79.5 270,79.5 270,319.5 0,319.5    0,349.5 460,349.5 460,200.5
" style="fill:#466289;" />
    <path d="M380,350.5l-10-60c-33.137,0-60,26.863-60,60c0,33.136,26.863,60,60,60L380,350.5z" id="XMLID_841_" style="fill:#233145;" />
    <path d="M370,290.5v120c33.137,0,60-26.863,60-60S403.137,290.5,370,290.5z" id="XMLID_842_" style="fill:#121923;" />
    <circle cx="370" cy="350.5" id="XMLID_843_" r="20" style="fill:#6B81A1;" />
    <path d="M89.999,290.5c-33.136,0-60,26.864-59.999,60c0,33.136,26.863,60,60,60l10-60   L89.999,290.5z" id="XMLID_844_"
style="fill:#354A67;" />
    <path d="M90,290.5v120c33.137,0,60-26.863,60-60C149.999,317.363,123.136,290.5,90,290.5z" id="XMLID_845_" style="fill:#233145;" /
>\n\t<circle cx="90" cy="350.5" id="XMLID_846_" r="20" style="fill:#8799B3;" />
  </g>
</svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Assess the object using the given SVG code, noting its semantic and geometric features. The corresponding SVG code: [SVG].

[Instruction]

The object in the code is a stylized representation of a truck. Here are the semantic and geometric characteristics:
Semantic Characteristics:
1. **Type of Object**: The object is a truck, which is a type of vehicle used for transporting goods.
2. **Function**: Trucks are typically used for carrying heavy loads over long distances.
3. **Components**: The truck has a cab where the driver sits and a cargo area for transporting goods.
4. **Color Scheme**: The truck is depicted with a blue body, a grey cargo area, and a yellow window, which are common colors used in illustrations to differentiate parts of the vehicle.

Geometric Characteristics:
1. **Shapes**:
  - **Rectangles**: The body of the truck, the cargo area, and the window are all represented using rectangular shapes.
  - **Circles**: The wheels of the truck are depicted as circles.
2. **Proportions**:
  - The cab is smaller compared to the cargo area, which is typical for trucks.
  - The wheels are proportionally smaller compared to the body of the truck.
3. **Lines and Angles**:
  - The truck is composed of straight lines and right angles, giving it a blocky, geometric appearance.
  - The wheels are perfect circles, indicating smooth, rounded edges.
4. **Symmetry**:
  - The truck has bilateral symmetry along its vertical axis, meaning the left and right sides are mirror code of each other.
5. **Perspective**:
  - The truck is shown in a side view, which is a common perspective for illustrating vehicles to clearly show their shape and structure.

These characteristics combine to create a simple, easily recognizable icon of a truck, often used in digital interfaces and illustrations.

[Response]

```
<svg id="Layer_1" style="enable-background:new 0 0 508 508;" version="1.1" viewBox="0 0 508 508" x="0px" y="0px" xml:space="preserve">
  <circle cx="254" cy="254" r="254" style="fill:#84DBFF;" />
  <polygon points="298,98.8 142,98.8 142,393.2 375.6,393.2 375.6,176.4 " style="fill:#FFFFFF;" />
  <polygon points="298,176.4 375.6,176.4 298,98.8 " style="fill:#E6E9EE;" />
  <g>
    <rect height="12.4" style="fill:#4CDBC4;" width="170.8" x="173.6" y="192.4" />
    <rect height="12.4" style="fill:#4CDBC4;" width="170.8" x="173.6" y="237.6" />\n\t<rect height="12.4" style="fill:#4CDBC4;" width="170.8"
x="173.6" y="283.2" />\n\t<rect height="12.4" style="fill:#4CDBC4;" width="170.8" x="173.6" y="328.4" />
  </g>
  <path d="M238.8,350c1.6-0.4,13.2-4.4,24-2.8c0-6,5.2-14.4,9.2-18.4l12.8-12.8l-6-6l-6-6L260,316.8
c-4,4-12.4,9.2-18.4,9.2C243.2,336.8,239.2,348.4,238.8,350z" style="fill:#FFD05B;" />
  <path d="M290.8,319.6c10.4-8.4,34.4-30.8,46-42.4c34.4-34.4,99.2-110.4,92-117.2c-6.8-6.8-82.8,58-117.2,92
c-11.6,11.6-34.4,35.6-42.4,46L290.8,319.6z" style="fill:#FF7058;" />
  <g>
    <polygon points="259.2,309.6 279.2,329.6 290.8,319.6 269.2,298.4  " style="fill:#324A5E;" />
    <circle cx="252" cy="337.2" r="2.4" style="fill:#324A5E;" />\n\t\n\t\t<rect height="18.4" style="fill:#324A5E;" transform="matrix(0.7071
0.7071 -0.7071 0.7071 314.9902 -72.8714)" width="0.8" x="245.059" y="334.591" />
    <circle cx="239.6" cy="349.6" r="1.6" style="fill:#324A5E;" />
  </g></svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Interpret the object using the provided SVG code, highlighting its semantics and geometric details. The corresponding SVG code: [SVG].

[Instruction]

The code displays a stylized icon representing a document with a pen. Here are the detailed characteristics:

1. **Semantic Characteristics:**
  - The icon symbolizes writing, editing, or authoring a document. It is commonly used in interfaces to represent actions such as creating a new document, signing, or editing text.\n   - The presence of the pen alongside the paper suggests activities related to writing, such as signing contracts or filling out forms.

2. **Geometric Characteristics:**
  - The document is depicted as a rectangle with rounded corners, enhancing its visual appeal and making it appear softer and more modern.
  - The lines inside the rectangle represent text lines, spaced evenly, suggesting a standard text document or form.
  - The pen, which is diagonally placed over the document, has a tapered shape typical of a fountain pen, indicating a formal or traditional approach to writing.
  - The overall composition is simple and uses flat colors and minimalistic design, typical of modern iconography used in digital interfaces.

3. **Color Scheme:**
  - The icon uses a light blue background which is calming and neutral, suitable for professional or educational contexts.
  - The document and pen are both white, which stands out against the blue background, ensuring good contrast and visibility.

This icon is effective in conveying its intended message due to its clear symbolism and straightforward design.

[Response]

```
<svg id="Layer_1" style="enable-background:new 0 0 512 512;" version="1.1" viewBox="0 0 512 512" x="0px" y="0px"
xml:space="preserve">
  <path d="M454.265,342.779L287.137,177.915l30.092-30.091c0.009-0.01,0.018-0.02,0.026-0.031
c33.735-33.764,33.726-88.678-0.026-122.43C300.876,9.007,279.13,0,256.001,0c-23.131,0.001-44.876,9.008-61.23,25.364
c-16.355,16.358-25.361,38.101-25.361,61.228c0,12.118,9.824,21.943,21.943,21.943s21.943-9.824,21.943-21.943
c0-11.406,4.443-22.13,12.509-30.198c8.068-8.068,18.792-12.509,30.199-12.51c11.407,0,22.131,4.443,30.199,12.509
c16.649,16.65,16.65,43.741,0.006,60.392c-0.003,0.001-0.004,0.003-0.006,0.004l-45.61,45.61L57.735,342.778
c-6.337,6.251-8.269,15.712-4.892,23.948c3.379,8.237,11.399,13.616,20.301,13.616h2.926h48.274h263.313v0.001h51.2
c8.903,0,16.922-5.379,20.301-13.615C462.534,358.492,460.602,349.031,454.265,342.779z
M255.999,208.844l129.367,127.615  l-258.733-0.001L255.999,208.844z" style="fill:#A9A8AE;" />
  <path d="M365.713,336.458H255.999H146.285c-12.118,0-21.943,9.825-21.943,21.943v21.941v109.715
c0,12.118,9.825,21.943,21.943,21.943h109.714h109.714c12.118,0,21.943-9.824,21.943-21.943V380.344v-21.943
C387.655,346.283,377.831,336.458,365.713,336.458z" style="fill:#FF6243;" />
  <path d="M146.285,336.458c-12.118,0-21.943,9.825-21.943,21.943v21.941v109.715
c0,12.118,9.824,21.943,21.943,21.943h109.714V336.458L146.285,336.458L146.285,336.458z" style="fill:#FF0C38;" />
</svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Based on the given SVG code, explain the object in terms of its semantics and geometric characteristics. The corresponding SVG code: [SVG].

[Instruction]

The code depicts a stylized icon of a clothes hanger. Here are the semantic and geometric properties of the object:
Semantic Properties:
1. **Representation**: The object is a clothes hanger, which is typically used for hanging clothes to keep them wrinkle-free.
2. **Function**: It symbolizes clothing, organization, and storage.
3. **Context**: This icon could be used in applications related to fashion, laundry, retail, or closet organization.

Geometric Properties:
1. **Shape**: The hanger is depicted with a simple, clean design. It has a triangular shape with a hook at the top.
2. **Color**: The hanger is primarily gray. Below the hanger, there is a square divided into two vertical halves. The left half is red, and the right half is orange.
3. **Lines and Angles**: The hanger consists of straight lines forming the triangular part and a curved line forming the hook. The square below has straight lines and right angles.
4. **Symmetry**: The hanger itself is symmetrical along the vertical axis, while the square below is also symmetrical along the vertical axis but divided into two different colors.
5. **Proportions**: The hook is proportionally smaller compared to the triangular part of the hanger. The square is smaller and centered below the hanger.

This icon is simple yet effective in conveying the concept of clothing and organization.

[Response]

```
<svg id="Capa_1" style="enable-background:new 0 0 450.493 450.493;" version="1.1" viewBox="0 0 450.493 450.493"
x="0px" y="0px" xml:space="preserve">
<path
d="M353.191,372.884h-17.686v-37.397c0-5.523-4.478-10-10-10h-24.971L283.61,127.998h12.218c5.522,0,10-4.477,10-10V
89h12.852
c5.522,0,10-4.477,10-10V10c0-5.523-4.478-10-10-10h-36.877c-5.522,0-10,4.477-10,10v20.771h-9.19V10c0-5.523-4.478-10-
10-10  h-54.724c-5.522,0-10,4.477-10,10v20.771h-9.189V10c0-5.523-4.478-10-10-10h-36.886c-5.522,0-10,4.477-10,10v69
c0,5.523,4.478,10,10,10h12.852v28.998c0,5.523,4.478,10,10,10h12.218l-16.924,197.489h-24.972c-5.522,0-10,4.477-10,10v3
7.397  H97.302c-5.522,0-10,4.477-10,10v57.609c0,5.523,4.478,10,10,10h255.89c5.522,0,10-4.477,10-10v-57.609
C363.191,377.361,358.714,372.884,353.191,372.884z M141.813,20h16.886v20.771c0,5.523,4.478,10,10,10h29.189
c5.522,0,10-4.477,10-10V20h34.724v20.771c0,5.523,4.478,10,10,10h29.19c5.522,0,10-4.477,10-10V20h16.877v49H141.81
3V20z   M164.665,89h121.163v18.998H164.665V89z
M186.957,127.998h76.579l16.925,197.489H170.033L186.957,127.998z M134.987,345.487
h180.519v27.397H134.987V345.487z M343.191,430.493h-235.89v-37.609h235.89V430.493z" />
</svg>
```

[SVG]

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request. Detail the object based on the SVG code given, with respect to its semantics and geometric features. The corresponding SVG code: [SVG].

[Instruction]

The object depicted in the code is a chess rook, also known as a castle. Here are its geometric and semantic properties:
Geometric Properties:
1. **Shape**: The rook has a rectangular base and a cylindrical body that tapers slightly as it rises. The top of the rook is characterized by a crenellated design, resembling the battlements of a castle.
2. **Symmetry**: The rook is symmetrical along its vertical axis.
3. **Lines and Angles**: The rook features straight lines and right angles, particularly in the crenellations at the top.
4. **Proportions**: The base is wider than the body, and the top is slightly wider than the body but narrower than the base.

Semantic Properties:
1. **Role in Chess**: In the game of chess, the rook is a major piece that moves horizontally or vertically across the board. It is one of the most powerful pieces due to its range of movement.
2. **Symbolism**: The rook symbolizes strength, stability, and protection, akin to a fortress or castle in medieval times.
3. **Position**: At the start of a chess game, each player has two rooks, positioned at the corners of the board.
4. **Notation**: In chess notation, the rook is represented by the letter "R".\n\nThe rook is an essential piece in chess strategy and is often involved in key moves such as castling, which involves the rook and the king.

[Response]

Figure 26: SVG examples of our symbolic instruction-following data.

E Details and More Results of Symbolic Instruction Tuning

E.1 Implementation Details

We use the *unsloth*[9] framwork to finetune the base models Llama3-8b-instruct and Gemma-1.1-7b-it. For both models, we use the exact same training setting: we finetune the base models with LoRA [36] on 1 NVIDIA H100 80GB gpu with learning rate 2e-4, batch size of 2 and for 1 epoch.

We use the *PEFT*[10] framework to test different fine-tuning methods when performing SIT. We choose two common fine-tuning methods LoRA [36] and orthogonal finetuning [75, 60] to fine-tune the base model Llama3.1-8b-Instruct. For both fine-tuning methods, we train on 8 NVIDIA H100 80GB gpus with learning rate 1e-4, per device batch size of 1 and for 1 epoch.

As introduced in Section 6, we also use *PEFT* to test if SIT can improve generic instruction tuning, by mixing our curated SIT data into the publicly available instruction tuning dataset *open-instruct*[11]. We use LoRA to fine-tune the base model Llama3.1-8b on 8 NVIDIA H100 80GB gpus with learning rate 1e-4, per device batch size of 1 and for 1 epoch. We test mixing with different SIT data splits, including 10K, 25K, 40K, 55K, and 72K. For example, Open-Instruct-SIT-10K, Open-Instruct-rev-SIT-10K and Open-Instruct-mixed-SIT-10K are constructed by mixing Open-Instruct with SIT-10K, rev-SIT-10K and mixed-SIT-10K. More specifically, rev-SIT-10K is constructed from SIT-10K according to Figure 12, while mixed-SIT-10K uniformly samples exactly 5K of the SIT instruction-following pairs and convert them to rev-SIT, while the rest 5K is kept unchanged. The best result is reported in the Table 3. We employ the widely-used *lm-evaluation-harness*[12] to obtain the results on a variety of LLM benchmarks.

[9] https://github.com/unslothai/unsloth

[10] https://github.com/huggingface/peft

[11] https://huggingface.co/datasets/VMware/open-instruct

[12] https://github.com/EleutherAI/lm-evaluation-harness

E.2 More Experiments in Symbolic Instruction Tuning

We additionally provide an ablation study of using different-size SIT data to finetune the base LLMs and measure their performance after SIT on the SGP-Bench. We uniformly sample 72k SVG programs from the *SVG Icons* dataset to build an instruction-following dataset using the text prompt examples in F.1 to query GPT-4v. The SIT-25k dataset is built by choosing the samples with the shortest code length out of the original 72k instruction following pairs. The SIT-10k dataset is a subset of SIT-25k, by uniformly sampling from the SIT-25k dataset. For SIT-40k and SIT-55k, we additionally sample more data with short code length from the *SVG Icons* dataset and mix it with SIT-25k. In this way, we can ensure that the smaller SIT dataset is always a subset of the bigger one.

We use the LLM-based evaluation because we noticed that after SIT, the generic instruction-following ability of the finetuned model degerates compared to the original model. We want to eliminate the cases where the finetuned model answers the questions correctly but do not follow the answer template, so that a matching-based evaluation will fail to extract meaningful answers. The results are shown in the Table 5. We notice that for Llama3-8B-instruct, the SIT will improve the generic semantic understanding up-to some SIT data size, afterwards the performance degenerates, while for Gemma-1.1-7b-it, the overall semantic understanding improves significantly without noticeable degeneration.

Dataset Size	Llama3-8B	Gemma-7B
Original	43.20	39.33
SIT-10k	**48.16** (+4.96)	**45.60** (+6.27)
SIT-25k	**51.43** (+8.23)	**46.87** (+7.54)
SIT-40k	**45.62** (+2.42)	**45.21** (+5.88)
SIT-55k	40.99 (-2.21)	**47.28** (+7.95)

Table 5: Ablation study of studying the effect of different-sized SIT data on the model's performance on the SPG-Bench (SVG-Understanding) using LLM-based answer extraction.

We also conducted an ablation study to determine whether the finetuning method affects the SIT results. Our findings indicate that the enhancement in the model's ability to understand symbolic programs is independent of the finetuning approach. Both OFT and LoRA significantly improve the model's understanding of symbolic programs, as shown in Table 6.

Dataset Size	LoRA	OFT
Llama 3.1-8B*	46.7	46.7
SIT-10k	**47.9** (+1.2)	**48.0** (+1.3)
SIT-25k	**49.8** (+3.1)	**50.3** (+3.6)
SIT-40k	**51.0** (+4.3)	**51.2** (+4.5)
SIT-55k	**51.3** (+4.6)	**51.4** (+4.7)

Table 6: Ablation study of studying the effect of different-sized SIT data on the model's performance on the SPG-Bench (SVG-Understanding) using LLM-based answer extraction and different fine-tuning methods. * The value differs from the value in the main table, because we use LLM-based evaluation to guarantee consistency.

F Text Prompt Template

F.1 Template for benchmark construction

We randomly sample from the following 20 prompts to generate the Symbolic Instruction Tuning (SIT) data:

```
"Describe in detail the semantic or geometric characteristics of
the object shown in the image."
"Offer a detailed description of the geometric or semantic
attributes of the object in this image."
"Can you provide a detailed account of the geometric or semantic
features of the object in the image?"
"Give a comprehensive description of the semantic or geometric
properties of the object depicted in the image."
"Elaborate on the geometric or semantic features of the object
in the image."
"Provide an in-depth description of the semantic or geometric
aspects of the object shown in the image."
"Detail the semantic or geometric features of the object in
the image."
"Explain in detail the semantic or geometric characteristics
of the object displayed in the image."
"Could you detail the geometric or semantic features of the object
in the image?"
"I need a detailed description of the geometric or semantic
attributes of the object in the image."
"Please describe the semantic or geometric features of the object
in the image comprehensively."
"Provide a thorough description of the geometric or semantic
properties of the object in this image."
"Can you elaborate on the semantic or geometric features of the
object in the image?"
"Describe precisely the semantic or geometric characteristics
of the object shown in the image."
"Give a detailed explanation of the geometric or semantic
features of the object in the image."
"Offer a complete description of the semantic or geometric
aspects of the object in the image."
"Detail the geometric or semantic properties of the object
depicted in the image."
"Explain the semantic or geometric features of the object in the
image in detail."
"Provide a detailed analysis of the geometric or semantic
features of the object in this image."
"Elaborate on the semantic and geometric characteristics of the
object shown in the image."
```

We use the following prompt to query GPT to construct the SGP-Bench (SVG) question-answer pairs.

```
Given the image, contruct in total 4 multiple-choice question-
answer pairs, with answer choices A, B, C, D, that concentrate
on the semantics or geometric features of the object in the img.
The first three questions are random.
The forth question should ask about the semantic of the
whole object.
Note: the format of the question-answer pairs should be
as follows:
===
Question: What is the capital of Germany?
Options: A) Rome; B) Beijing; C) Paris; D) Berlin
Answer: D
===
Question: What is the color of the sky on a clear day?
Options: A) Gray; B) Blue; C) Orange; D) Green
Answer: B
===
```

We use the following prompt to query GPT to construct the SGP-Bench (CAD) question-answer pairs:

```
Construct five multiple-choice question-answer pairs, with
answer choices A, B, C, D, that concentrate on the geometry
of the CAD object in the image.
Note: the format of the question-answer pairs should be
as follows:
===
Question: How did Spider-Man get his powers?
Options: A) Bitten by a radioactive spider; B) Born with them;
C) Military experiment gone awry; D) Woke up with them after
a strange dream
Answer: D
===
Question: What is the color of the sky on a clear day?
Options: A) Gray; B) Blue; C) Orange; D) Green
Answer: B
===
```

We randomly sample from the following 20 prompts to construct the questions for the SGP-MNIST benchmark. We do not need to query GPT because we have the ground truth label for every SGP-MNIST code.

```
"What number between 0 and 9 is shown in this picture?"
"Identify the digit from 0 to 9 depicted in this image."
"Which number from 0 through 9 is illustrated in this image?"
"Can you tell which digit (0-9) this image represents?"
"What is the digit, from 0 to 9, that appears in this image?"
"Determine the digit between 0 and 9 displayed in this image."
"Spot the digit (0-9) that this image portrays."
"Which of the digits 0-9 does this image illustrate?"
"Recognize the digit from 0-9 shown in this picture."
"From 0 to 9, what digit is shown here in this image?"
"What single digit from 0-9 is presented in this image?"
"Specify the digit (0-9) that is represented by this image."
"What digit, ranging from 0 to 9, does this image show?"
"Identify which one of the digits 0-9 is depicted in this img."
```

```
"Name the digit between 0 and 9 that this image represents."
"Which digit, 0 through 9, is displayed in this picture?"
"Tell which digit from 0 to 9 is shown in this image."
"Pinpoint the digit from 0-9 represented in this image."
"What digit from the range 0-9 is depicted in this image?"
"Indicate which digit (0-9) is illustrated in this image."
```

Given the question-answer pairs, either through querying GPT (SVG) or randomly sample from a pre-defined prompt pool, we use the following question template to construct the questions for our SGP-Bench (SVG):

```
Examine the following SVG code carefully and answer the
question based on your interpretation of the rendered image.

{SVG}

Question: {Question}
```

Given the question-answer pairs we use the following question template to construct the questions for our SGP-Bench (CAD):

```
Examine the following CAD code carefully to understand the
3D object it generates and answer the question based on your
interpretation of the rendered image of that object.

{CAD}

Hint: the CAD code has the following syntax:

{Hint}

Question: {Question}
```

When constructing the SGP-Bench (CAD), we also provide the syntax of CAD code:

```
CAD code consists of a sequence of CAD commands that describe
a 3D object.
The commands fall into two categories: sketch and extrusion.
Sketch commands are used to specify closed curves on a 2D plane
in 3D space. Each closed curve is referred as a loop, and one
or more loops form a closed region called a profile. A loop
always starts with an indicator command <SOL> followed by a
series of curve commands. All the curves on the loop are in
counterclockwise order, beginning with the curve whose starting
point is at the most bottom-left.
In total, there are three possible curve commands: Line, Arc,
and Circle.
Line(x, y): a line, with x, y as line end-point.
Arc(x, y, u, f): an arc, with x,y as arc end-point, u as sweep
angle and f as whether it is counter-clockwise, f=0 means it is
counter-clockwise, f=1 means it is not counter-clockwise.
Circle(x, y, r): a circle, with x,y as the center point and r
as the radius.
The extrusion command has two purposes:
1) It extrudes a sketch profile from a 2D plane into a 3D body,
and the extrusion type can be either one-sided, symmetric,
or two-sided with respect to the profile’s sketch plane.
2) The command also specifies (through the parameter b in Ext)
how to merge the newly extruded 3D body with the previously
```

```
created shape by one of the boolean operations: either creating
a new body, or joining, cutting, or intersecting with the
existing body.
Ext(x, y, z, o, p, q, s, e, f, b, u): extrude operation, with
x, y, z as the sketch plane orientation, o, p, q as the sketch
plane origin, s as the scale of the associated sketch profile,
e, f as the extrude distances towards both sides, b as the type
of merge operation (could be New-body operation, join operation,
cut operation and intersect operation) and u as the extrude
type (could be one-sided, symmetric or two-sided).
<EOS> means the end of the code.
```

```
CAD code consists of a sequence of CAD commands that describe a
3D object.
The commands fall into two categories: sketch and extrusion.
Sketch commands are used to specify closed curves on a 2D plane
in 3D space. Each closed curve is referred as a loop, and one
or more loops form a closed region called a profile. A loop
always starts with an indicator command LOOP followed by a
series of curve commands.
Possible primitive types are defined with the following
parameters:
Arc(start_point,end_point,center_point,radius,normal,
start_angle,end_angle,reference_vector),
Circle(center_point,radius,normal), Line(start_point,
end_point), NurbsCurve(degree,knots,rational,
control_points,weights,periodic), Ellipse(major_axis,
major_axis_radius,minor_axis_radius,center_point,normal),
EllipticalArc(major_axis,major_axis_radius,
minor_axis_radius,center_point,normal).
The extrusion command ExtrudeFeature(operation, start_extent,
extent_type, extent_one, extent_two) has two purposes:
1) It extrudes a sketch profile from a 2D plane into a 3D body,
and the extrusion operation can be either one-sided, symmetric,
or two-sided with respect to the profile’s sketch plane.
2) The command also specifies (extent_type) how to merge the
newly extruded 3D body with the previously created shape by one
of the boolean operations: either creating a new body, or
joining, cutting or intersecting with the existing body.
```

```
CAD code consists of a sequence of CAD commands that describe
a 2D object.
The commands fall into two categories: primitive and constraint
In total, there are five possible primitive types:
Point(x, y), Line(dirX, dirY, pntX, pntY, startParam, endParam)
Circle(xCenter, yCenter, xDir, yDir, radius, clockwise),
Arc(xCenter, yCenter, xDir, yDir, radius, clockwise, startParam
endParam), and Ellipse(xCenter, yCenter, xDir, yDir, radius,
minorRadius, clockwise).
x, y: the point coordinates.
dirX, dirY: the unit direction vector.
xCenter, yCenter: the coordinates the center point.
clockwise: a boolean value that indicates the orientation of
the unit direction vector.
pntX, pntY (Line): the coordinates of a point on the line.
startParam, endParam (Line): signed start/end point distances
to the point (pntX, pntY).
startParam, endParam (Arc): start/end angles to the unit
```

```
direction vector.
All primitives have an isConstruction boolean parameter
indicating if a primitive is to be physically realized or
simply serve as a reference for other primitives.
All constraints act on at least one primitive, indicated by the
corresponding number.
```

F.2 TEMPLATE FOR EVALUATING MODELS ON SGP-BENCH

When evaluating different models on our SGP-Bench, we use the following evaluation template (multiple choice):

```
Answer the following multiple choice question. The last line of
your response should be of the following format:
'Answer: $LETTER' (without quotes) where LETTER is one of ABCD.
Think step by step before answering.

{Question}

A) {A}
B) {B}
C) {C}
D) {D}

Important, the last line of your response must be of the
following format:
'Answer: $LETTER' (without quotes) where LETTER must be one of
A, B, C or D.
```

When evaluating different models on our SGP-Bench, we use the following evaluation template (generation):

```
Solve the following problem step by step. The last line of your
response should be of the form Answer:
$ANSWER (without quotes) where $ANSWER is the answer to the
problem.

{Question}

Important, put your answer on its own line after "Answer:",
and you do not need to use a \\boxed command.
```

We use the following template, when we perform LLM-based evaluation:

```
Please read the following example. Then extract the answer
from the model response and type it at the end of the prompt.

Hint: The last line of your response should be of the following
format: 'Answer: $LETTER' (without quotes) where LETTER is
one of ABCD.
Question: What is the primary color of the object in the image?

A) Red
B) Blue
C) Black
D) Green

Model response: **Step 1: Examine the image**\n\nThe image
```

```
consists of various shapes filled with different colors.
We need to identify the primary color of the object in the
image.\n\n**Step 2: Focus on the dominant color**\n\nThe shapes
that cover the largest area in the object are filled with
shades of blue and its variations.\n\n**Answer: B**

Extracted answer: B

Hint: The last line of your response should be of the following
format: ’Answer: $LETTER’ (without quotes) where LETTER is
one of ABCD.
Question: What is the background color of the image?

A) Red
B) Green
C) Blue
D) Yellow

Model response: Answer: The background color is blue.

Extracted answer: C

Hint: The last line of your response should be of the following
format: ’Answer: $LETTER’ (without quotes) where LETTER is
one of ABCD.
Question: What is the shape of the buckle on the object?

A) Circle
B) Triangle
C) Square
D) Hexagon

Model response: Answer: D) Hexagon.

Extracted answer: D

Hint: The last line of your response should be of the following
format: ’Answer: $LETTER’ (without quotes) where LETTER is
one of ABCD.
Question: What type of object is shown in the image?

A) Watch
B) Belt
C) Bracelet
D) Necklace

Model response: The object in the code is a watch.

Extracted answer: A

Hint: The last line of your response should be of the following
format: ’Answer: $LETTER’ (without quotes) where LETTER is
one of ABCD.
Question: What is the primary color of the object in the image?

A) Blue
B) Yellow
C) Green
D) Red
```

```
Model response: The primary color of the object in the code
is yellow.

Extracted answer: B
```