

Document status: **Final**

Last update: Apr 21, 2024

[Standard Eval] - **Coding - Task** **Attempter Guide**

A strict reminder that the usage of AI Assistants/External Tools to generate and reproduce written texts, e.g. justifications, is strictly prohibited. Hereby you're permissioned to use AI for auxiliary purposes, such as generating boilerplate for code testing. You still have to run the code yourselves, and cannot rely on AI to run the code for you. If you are found to be misusing any AI Assistants, you will be removed from the project.

Important: Including a technical justification pinpointing all issues found is mandatory. **Always** state your response preference and back it up with evidence found when thoroughly testing the response code and evaluating response correctness/prompt adherence and other relevant dimensions.

Crucial: Do not penalize responses that contain code not formatted in markdown. This is most likely due to faulty response scraping, rather than an error in the model's response. The original code in the response was indeed formatted in markdown. Instead, **flag and skip** those tasks so we can reupload them and correct the markdown formatting accordingly

Please refer to [Standard Eval - High Quality Guide](#) for high quality task examples.

Abstract

Welcome to the [Standard Eval] Code Project! Your participation in this project signifies your commitment to excellence and your dedication to understanding AI models. By participating, you are actively contributing to the establishment of fair evaluation between state-of-the-art LLM models across important dimensions, thereby making significant strides in the development of Artificial intelligence.

Throughout this training, you will learn how to effectively evaluate responses generated by different LLMs, identify strengths and weaknesses, and provide insightful justifications for your evaluation.

This document functions as your go-to guideline for Standard Eval Code Attempts. Please read this document thoroughly to get started. We wish you good luck and thank you in advance for your contributions to the [Standard Eval] Code Eval project.

Content

[Section 1 - Overview](#)

[1.1 Task Overview](#)

[1.2 Dimension Evaluation Table](#)

[1.3 Workflow Overview](#)

[Section 2 - Workflow Walkthrough](#)

[2.1 Assess the Programming Language](#)

[To skip the task, simply click on the three dots located in the top right corner.](#)

[2.2 Review the Prompt](#)

[2.2a Prompt Examples](#)

[2.3 Analyze Model Responses](#)

[2.4 Answer the questions for each model response](#)

[2.4a Code Execution](#)

[2.4b Prompt Adherence / Understanding](#)

[2.4c Correctness / Functionality](#)

[2.4d Performance / Efficiency](#)

[2.4e Readability / Documentation](#)

[2.5 Answer Questions for the SxS \(side-by-side\) model rating](#)

[2.5a Side-by-side model rating](#)

[2.5b SxS Score](#)

[2.5c Justification](#)

[Section 3 - Task Examples](#)

[3.1 Cheat Sheet](#)

Section 1 - Overview

1.1 Task Overview

You will be provided with a prompt along with two model responses. Your tasks include:

1. **Evaluating the prompt:** Is the prompt clear or ambiguous?
2. **Executing the code:** Are you able to execute the code in the integrated IDE or a local IDE? Include the output/error in the std_out. Please do your absolute best to run the code.
3. **Evaluating each model response based on four dimensions:**
 - a. Prompt Adherence / Understanding
 - b. Correctness / Functionality
 - c. Performance / Efficiency
 - d. Readability/ Documentation
4. **Comparing the model responses:**
 - a. Perform a Side by Side evaluation of both responses and indicate a response preference using the SxS score. This is critical for the project to succeed. Please ensure thorough analysis and consideration of the four dimensions when choosing the better response.
 - b. Select the correct SxS score based on the difference in response quality.
 - c. Write a detailed justification to explain/justify the preference ranking.

1.2 Dimension Evaluation Table

Dimension	Required	Description
Prompt Adherence / Understanding	Always	The model adheres to the prompt and understands all its requests.
Correctness / Functionality	Always	The code executes without errors (Code Execution Correctness) and produces the correct output (Code Output Correctness) based on the intent of the model . Also covers factuality of all the written text and claims made - including inline code comments.
Performance / Efficiency	When Code is present	The code runs without any performance concerns. It follows good practices resulting in efficiency.
Readability / Documentation	Always	<p>The response has the necessary documentation helping to understand the code. The code is readable due to good formatting and utilization of mnemonic variable and function names.</p> <p>The written explanation is well-structured and visually organized. Employing whitespace, separate lines, and appropriate formatting such as bolding for emphasis where applicable.</p>

1.3 Workflow Overview

Follow the below workflow to execute the outlined task efficiently:


- Assess the programming language: Skip the task if you are uncomfortable with a specific programming language or Computer Science domain.
- Review the prompt: Familiarize yourself with all requirements/constraints specified in the prompt. Findings are used to assess Prompt Adherence / Understanding.

- Individually analyze the model responses: Thoroughly examine both model responses provided by reading the response, executing the code and rating the response against the 4 dimensions.
 - Answers questions for the individual model responses.
 - Execute any response code provided and answer the execution questions. Include the `std_out`, which may consist of either a standard error or a standard output.
 - Answer "No" to the question "Does the response answer the prompt correctly...?" if there are any issues in the below four dimensions:
 1. Prompt Adherence / Understanding
 2. Correctness / Functionality
 3. Efficiency / Performance
 4. Readability / Documentation
 - **Side note:** A text box should pop up after answering "No". You are required to meticulously document all issues found.
 - Answer questions for the side-by-side model preference rating based on the difference in response quality, provide justifications, and save your responses.
 - Submit the task
 - Ensure that you select "No" for "Does the response answer the prompt correctly in terms of results, instructions, and executability" (when coding is required)?" if you identify any issues.
 - Expect to find at least one minor issue most of the time. We're actively improving and training an imperfect model. **Therefore, it's critical that we do a fair evaluation and penalize the model when necessary.**
-

Section 2 - Workflow Walkthrough


2.1 Assess the Programming Language

If you are uncomfortable with a specific programming language, you are motivated to skip the task. Only work on languages you're highly proficient in.

**Short Instructions**
Follow these instructions when tasking





LANGUAGE: C++

- If you are not proficient in the language SKIP the task
- Read the Prompt
- Read the first Model Response and answer the questions
- Read the second Model Response and answer the questions
- Answer the questions about the Prompt
- Make a comparison between both Model Responses





write a c++ program Write a function that reverses a string. The input string is given as an array of characters s. You must do this by modifying the input array in-place with O(1) extra memory. Example 1: Input: s = ["h","e","l","l","o"] Output: ["o","l","l","e","h"] Example 2: Input: s = ["H","a","n","n","a","h"] Output: ["h","a","n","n","a","H"] Constraints: 1 <= s.length <= 105 s[i] is a printable ascii character. class Solution { public: void reverseString(vector& s) {} };

To skip the task, simply click on the three dots located in the top right corner.




Submit Task

 Report Sensitive Content

 Skip Task


2.2 Review the Prompt

Thoroughly read the prompt and familiarize yourself with all requirements specified in the prompt. This is used to evaluate Prompt Adherence of the response.

**Short Instructions**
Follow these instructions when tasking

LANGUAGE: C++

- If you are not proficient in the language SKIP the task
- Read the Prompt
- Read the first Model Response and answer the questions
- Read the second Model Response and answer the questions
- Answer the questions about the Prompt
- Make a comparison between both Model Responses



write a c++ program Write a function that reverses a string. The input string is given as an array of characters s. You must do this by modifying the input array in-place with O(1) extra memory. Example 1: Input: s = ["h","e","l","l","o"] Output: ["o","l","l","e","h"] Example 2: Input: s = ["H","a","n","n","a","h"] Output: ["h","a","n","n","a","H"] Constraints: 1 <= s.length <= 105 s[i] is a printable ascii character. class Solution { public: void reverseString(vector& s) {} };

Then, assess if the prompt/request is either clear or ambiguous.

Is the prompt clear? *

☐ Yes, the request is clear.

☐ No, the request is ambiguous.

2.2a Prompt Examples

Ambiguous Prompt/Request

Can you modify the Python function to process data?

- Clarity of intent: The request could be interpreted as asking if the listener has the capability to modify a Python function to process data, likely resulting in a 'yes' or 'no' answer. Alternatively and more probable, it could be understood as an implicit request to actually modify a Python function.
- Self-contained: The prompt is not self-contained, as it does not specify which Python function needs modification.
- Specificity: The method of data processing is left to the models interpretation, as the prompt does not state a clear processing method. However, this is considered acceptable if there is enough context within the code for the model to suggest and implement a relevant data processing method.

Clear Prompt/Request

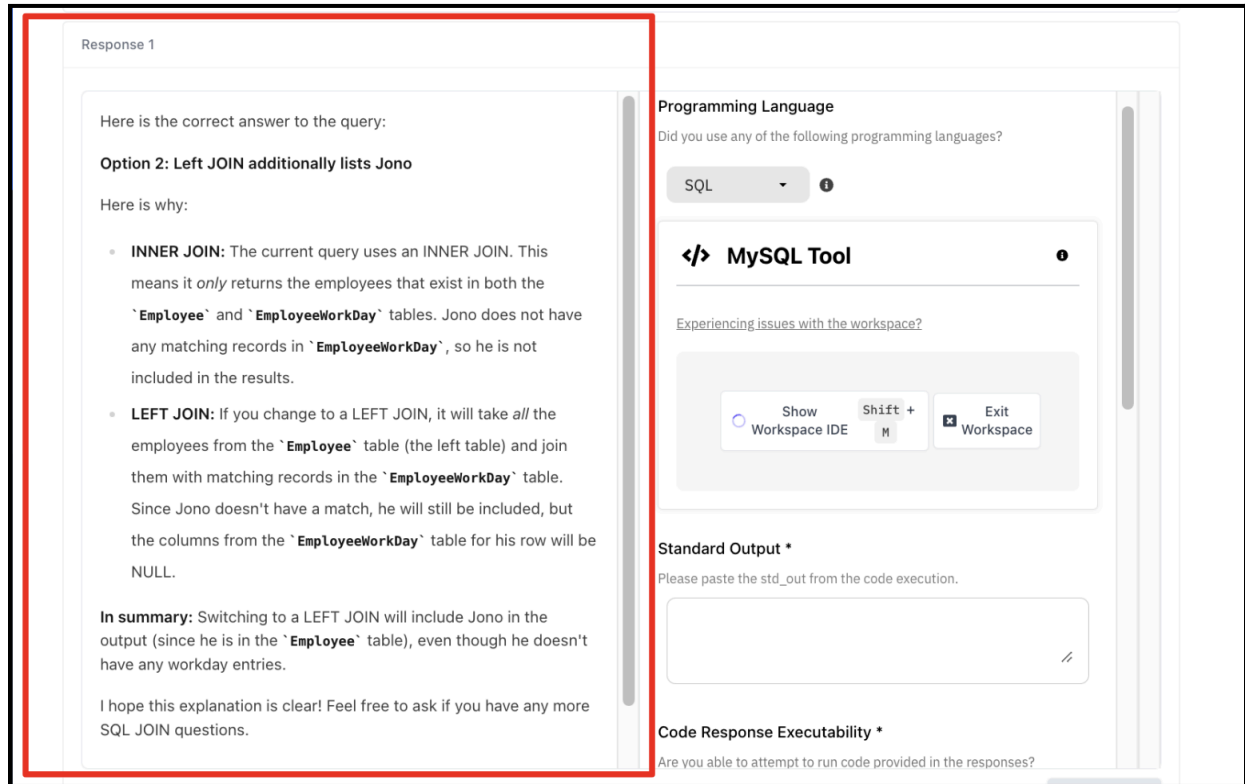
Create a Python program that reads sales data from a CSV file and calculates the top 3 products in terms of total sales. The CSV file consists of a product_id, product_name, quantity_sold and price_per_unit.

- Clarity of intent: The prompt has a clear request/intent to create a Python program with a specific functionality.
- Self contained: The prompt is self contained and does not depend on sources outside of the prompt context.
- Specificity: It mentions how the data should be processed and the structure of the data.

2.3 Analyze Model Responses

Thoroughly examine both model responses provided.

- There will be one model response (Response 1), and then there will be a second model response (Response 2) upon scrolling down.



2.4 Answer the questions for each model response

You will be asked to answer the following questions for each model:

1) Q: Is there code present? Please focus solely on the code provided in the response when answering this question. Disregard any code present in the prompt.

- ☐ Yes
☐ No

1.1) Q: Programming Language - Did you use any of the following programming languages?
[Dropdown box]

1.2) Q: Which of the following <language> workspaces best applies to your code?
[Dropdown box]

1.3) Q: Were you able to run the code in the IDE in this tool?

☐ Yes

☐ No, but I was able to run this code with my own IDE

1.4) Q: Standard Output - Please paste the std_out from the code execution.

2) Q: Is the response free from any issues in the four dimensions?

☐ Yes

☐ No

... Flag the issues found in Question 2.1. It's multi-select so you can select multiple issues. // **Not flagging issues here but mentioning them in the justification is considered an incoherent task and results in project removal.**

☐ I cannot verify - Avoid selecting "I cannot verify" unless absolutely necessary

2.1) Q: What are the main issues with this response? - Allows multi-select

☐ Prompt Adherence / Understanding

☐ Correctness / Functionality

☐ Performance / Efficiency

☐ Readability / Documentation

2.2) Explanation // **Include all issues found**


[Text box]

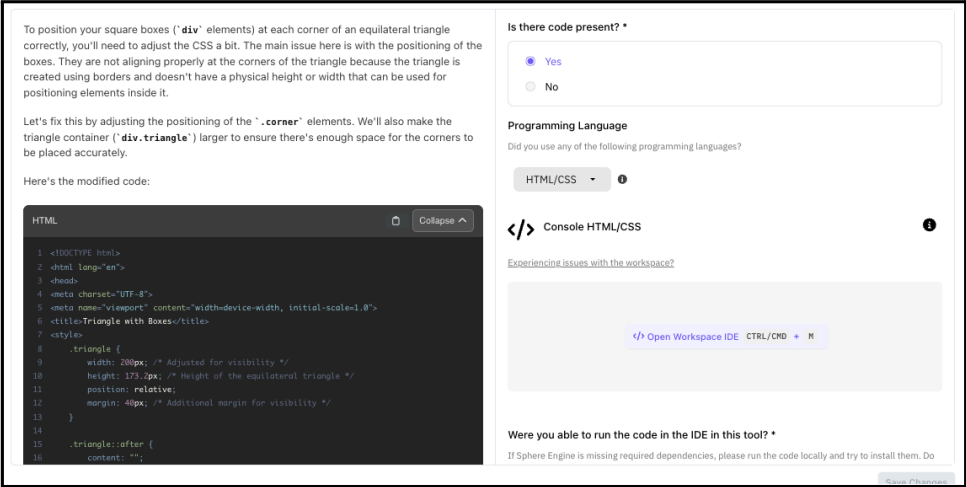
2.4a Code Execution and std_out

Focus area: Were you able to execute the code and test it? What language did you use, and what is the std_out? This only applies to responses that contain code. If there is code in the response, you're **always** required to run that code in either the provided Workspace IDE or your local IDE. Answer question 1.1 to 1.4

- 1) If multiple code snippets are included with each different output, the std_out includes different outputs of multiple snippets.

- 2) If there are multiple code snippets that form a single program you will bundle the code snippets together and execute the main program and include the output.
- 3) If the code relies on auxiliary code such as but not limited to helper functions, SQL tables, modules, etc, create the necessary file structure to ensure it runs (limit to 5 minutes). Parse the std_out from the code execution in the appropriate environment.
 - Ensure any required SQL tables are included if created.
 - Alternatively, if the code cannot be verified, select "I cannot verify" and explain the missing components needed to make the code functional.
- 4) If dependencies are missing in the integrated IDE you must use your local IDE to run the code.
- 5) If the code does not have any output, please supplement it with tests to generate output where feasible. For example, you can use the built-in print method in Python or console.log in JavaScript.
- 6) If the code produces an error you should include the error and attempt to resolve the issue within 5 minutes. Be aware that you do not actually fix the model response and should still rank it as having issues.
 - **Mandatory:** Always include the std_out of the original code.
 - **Optional:** You may include the std_out of the corrected code below the std_out of the original code. This helps in evaluating the response's code quality.
- 7) If the result of the code execution is a visual output such as an HTML web page or a plotted graph rather than text-based, the std_out should be a description of the content displayed. This description should cover the page layout and functionality, graph trend, titles, and axis labels, among other relevant elements. For further reference see std_out of this Python program that results in a graph plot:

 [Standard Eval - High Quality Guide](#) .
- 8) If no code is present in the response, the std_out is "N.A." Exercise caution when responding with "N.A.," as it's considered low effort if code is provided in the model's response.
 - Select "no" for the question "Is there code present?"
 - **Important:** Some model responses may include code not formatted in markdown, which still counts as code presence. See this example: // TBC



2.4b Prompt Adherence / Understanding

Focus area: The extent to which the model adheres to the prompt and understands all its requests/constraints. Avoid considering the correctness of the response when answering this question; this will be accounted for in the correctness checkbox instead (e.g. when there is faulty code provided). **The prompt can fully adhere to the prompt but have correctness issues such as seen in this example** [Standard Eval - High Quality Guide](#)

Rubric

Answer choice	Definition
Yes	<ul style="list-style-type: none">The response fully adheres to all the prompt’s requirements/constraints.

No	<ul style="list-style-type: none"> • The response exhibits issues in Prompt Adherence as key components of the prompt are overlooked. <ul style="list-style-type: none"> ◦ Example #1: The response uses a different code pattern than what the prompt specified, yet it is considered correct. <ul style="list-style-type: none"> ■ Prompt: “Generate a python script that prints out the fibonacci number based on the index number entered using recursion.” ■ Response: The response provided prints out the nth fibonacci number using iteration. Response is still fully correct/functional as it does not claim to use recursion. For further reference check this example: Standard Eval - High Quality Guide ◦ Example #2: It adheres to the main prompt request but then proceeds to do more than asked for. <ul style="list-style-type: none"> ■ Prompt: “How do I plot a graph in matplotlib?” ■ Response: “Understands the main request of plotting a graph but overdoes it by creating a 3D graph and customizing the axes. A simpler example was expected. The response is overcomplicated.” For further reference check this example: Standard Eval - High Quality Guide ◦ Example #3: It solely outlines the steps for writing the code when the prompt implicitly/explicitly requests to generate the code: Example TBC
----	--

2.4c Correctness / Functionality

Focus area: Is the response effective, correct and factual? Does the program work as intended? Check that all code executes without errors - code execution correctness. The code/program must produce the correct output based on the model's intent - code output correctness. Verify factuality/accuracy of all written text. This includes code comments and the text surrounding the code snippets. **Critical: The model can misunderstand/not adhere to the prompt request but still produce a correct/functional program that works as intended by the model. In which, the response should be marked as having no issues in Correctness but flagged for Prompt Adherence issues.**

NOTES:

- Do not penalize the model for missing context that is out of the prompt scope such as for example API routes. It was perhaps not up to the model to create those routes.
- **Important:** If the original code fails to execute and you opt to address code issues to test the output and assess the original code's quality, the response should still be rated as having correctness issues.

Rubric

Answer choice	Definition
Yes	<ul style="list-style-type: none"> • The code executes without errors (Code Execution Correctness) and generates an output that aligns precisely with the model intent (Code Output Correctness) • The code is safe, free of vulnerabilities and follows best practices. • All written text (including inline code comments) is factual and accurate.
No	<ul style="list-style-type: none"> • The response exhibits major issues in correctness: <ul style="list-style-type: none"> ◦ Code does not execute due to flawed logic - code execution correctness ◦ The output produced by the code does not align with the program intent - code output correctness ◦ Safety issues ◦ False claims • The response exhibits minor issues in correctness: <ul style="list-style-type: none"> ◦ Syntax issues such as variable names or methods being misspelled that prevents compilation - issue in code execution correctness. ◦ A missing important statement or a Java class without a main method- code execution correctness ◦ Python code with indentation issues - code execution correctness. Also penalized in readability.

2.4d Performance / Efficiency

Focus area: Do the used methods/code patterns result in acceptable performance and efficiency? This evaluation should check if the code is free from any major performance drawbacks or inefficiencies which could be avoided.

Rubric

Answer choice	Definition
Yes	<ul style="list-style-type: none"> • The response does not have any performance or efficiency concerns. <ul style="list-style-type: none"> ◦ If a more efficient option exists but it would be much more complex and the best performance is not being requested by the prompt, mark it as correct. Performance needs to be at least adequate.
No	<ul style="list-style-type: none"> • The code exhibits performance/efficiency issues. Examples: <ul style="list-style-type: none"> ◦ Unnecessarily slow code, i.e. code that runs on $O(n^2)$ but could run on $O(n \log n)$ ◦ The code uses global variables unnecessarily ◦ The code loads into memory the full dataset, which could be large, when it could instead use lazy load and process data as needed, saving memory and speeding up processing time

	<ul style="list-style-type: none"> ○ The query uses a union join which performs poorly with large data, when it could have used union all instead ○ Stack overflows or infinite loops. Also penalized in correctness as the code non-functional.
--	--

2.4e Readability / Documentation

Focus area: Is the necessary documentation present and is the response of the correct length? Is the response well written and is the code readable due to proper formatting?

Check if the necessary response documentation is present and if the response is adequate in length. Documentation covers both inline code comments and the written text surrounding the code. Formatting issues that reduce code readability are also penalized in this dimension.

Rubric

Answer choice	Definition
Yes	<ul style="list-style-type: none"> ● The code (if present) has an adequate amount of documentation in it. ● The written text surrounding the code snippets is well written / structured. ● The response uses appropriate variable names / function names that enhance readability. ● The response is of appropriate length. ● The response remains a neutral tone.
No	<ul style="list-style-type: none"> ● The response has inadequate/missing documentation and is considered too short ● The response does not highlight the code because the programming language is not specified in the markdown. ● The response is overly verbose. <ul style="list-style-type: none"> ○ The response contains unnecessary/overcomplicated content ○ The response contains repetitive content ● The code is not readable due to bad formatting. <ul style="list-style-type: none"> ○ Critical: If the code is also not executable due to bad formatting such as in Python, the response is also considered to have issues in Correctness. ● The response diverges from the expected tone <ul style="list-style-type: none"> ○ As an AI model it should not make use of subjective language and remain a neutral tone. The model cannot hope nor feel. ● The response is poorly structured or the code has bad variable names - using mnemonic variables is best practice.

2.5 Answer Questions for the SxS (side-by-side) model rating

2.5a Side-by-side model rating

Compare the two model responses, and indicate whether the model response 1 was better or the model response 2 was better on a 1-7 scale:

Select the best response

Where 1 means response 1 is much better, 7 means response 2 is much better, and 4 is neutral.

1

2

3

4

5

6

7

Response 1 is much better than
Response 2

Both Model Responses are of the
same quality

Response 2 is much better
Response 1

Then answer the following question:

Is one response objectively better than the other? *

☒ Yes

☐ No

2.5b SxS Score

Use the following rubric to select the correct SxS score and asses if one response is either objectively or subjectively better than the other:

Rubric

Response 1 is Much Better	Response 1 is Better	Response 1 is Slightly Better	Neutral	Response 2 is Slightly Better	Response 2 is Better	Response 2 is Much Better
------------------------------	-------------------------	----------------------------------	---------	----------------------------------	-------------------------	------------------------------

- Scores of 1, 2, 6, and 7 indicate an objective preference and a large response quality difference -> Select “Yes” for “Is one response objectively better than the other?”
 - Select a score of 1 or 7 if one response is flawless whereas the other response has substantial issues.
 - Select a score of 2 or 6 if both responses have issues but one response is substantially better.
 - Select a score of 2 or 6 if one response is flawless where the other response has minimal issues.
- Scores of 3, and 5 indicate a subjective/personal preference and a minimal response quality difference -> Select “No” for “Is one response objectively better than the other?”
- A score of 4 indicates a neutral preference. Both responses are alike in both approach/content and quality -> Select “No for “Is one response objectively better than the other?”
- Side note: **Use your best judgment when selecting a SxS score.**

Pro Tip:

When evaluating model responses, consider both the Code Execution Correctness and Code Output Correctness together. Review the following scenarios for clarity:

- **Scenario A:** If both model responses adhere to the prompt and the codes compile successfully, assess whether one of them produces incorrect output. In this case, the response with the correct code output is preferred.
- **Scenario B:** If both model responses adhere to the prompt but one model response code executes while the other does not, it's possible that the non-executing code contains a syntax error but would logically produce the correct output. Conversely, the executing code may have the correct syntax but lacks correct output due to a logical mistake. Logical errors should be evaluated as major issues under Correctness while syntax errors should be evaluated as minor issues under Correctness, making the response with the syntax error preferable in this scenario.

2.5c Justification

Once you've chosen a side-by-side model score, you'll be prompted to provide a justification. Please aim to encompass all aspects of your thought process, as this aids in understanding the strengths and weaknesses of the models. This insight is valuable for devising strategies to enhance model performance across various applications.

Critical: Make sure that your justification aligns with the dimension issues flagged, SxS score and std_out. Incoherent tasks are considered spam. Annotate all found issues and meticulously compare both responses justifying your SxS score.

✓ Here are some examples of a good Side by Side choice justification for this task:

@Response 1 is better out of the two responses. It meets most of the prompt requirements, including (1) verifying correctness by running test cases and (2) handling edge cases gracefully. However, it falls short in implementing code optimization techniques, such as using list comprehension for efficient iteration. In contrast, @Response 2 fails to provide any code, which goes against the prompt's instruction to generate robust and optimized code.

In comparing @Response 1 and @Response 2, @Response 1 is better. It demonstrates proficiency in (1) utilizing appropriate data structures and (2) optimizing algorithmic efficiency. Nonetheless, it overlooks error handling mechanisms, which may lead to unexpected runtime behavior. Conversely, @Response 2 lacks depth, failing to implement key functionalities outlined in the prompt; for example: create a table that stores employee information.

✗ Here are some examples of a bad Side by Side choice justification for this task:

@Response 1 is better than @Response 2 because the code output is correct.

Both responses followed the prompt.

@Response 1 is better out of the two responses. It meets most of the prompt requirements except for one. In contrast, @Response 2 fails to provide any code.

Section 3 - Task Examples

Please refer to [☰ Standard Eval - High Quality Guide](#) for high quality task examples.

3.1 Cheat Sheet

Dimension Cheat Sheet

1. Prompt Adherence / Understanding: Does it adhere to the prompt and address all its requirements?

- **YES:** The response adheres to the prompt and addresses all requirements.
- **NO:** The response solely outlines the steps for writing the code when the prompt implicitly or explicitly requests to generate actual code.
- **NO:** The response correctly implements the functionality but deviates from the specified use case.
- **NO:** The response answers the prompt but then proceeds to produce overly complicated/unasked for content. It does more than asked.

2. Correctness / Functionality: Is the program effective and executes as intended and stated by the model? Is all written text factual? Is the code safe and free of vulnerabilities?

- **YES:** The program runs without errors and works as intended and stated by the model. The response is factual overall. You might have to provide tables, write code, create a supporting file structure etc, this is acceptable.
- **NO:** The code requires some edits/fixes to function correctly. Code could be either broken or incomplete as it is e.g. missing helper functions within the scope of the requested program.
- **NO:** The code has vulnerabilities that were not pointed out.

3. Performance / Efficiency: Are the code patterns/methods used efficient and good practice? Does the code perform well?

- **YES:** The response is scalable or performs well enough for what was asked.
- **NO:** The code performs poorly with large data and should be improved.

4. Readability / Documentation: Is the response well structured and is necessary documentation present? Is the code properly formatted and are all variables/functions well named?

- **YES:** The code is properly formatted and the necessary response documentation is present.
- **NO:** It has poorly named variables, functions, classes, methods etc.
- **NO:** It fails to provide explanation/documentation which was of crucial importance.
- **NO:** The explanation is too lengthy considering the request making the response overly verbose.