Project Blackhat Instructions

Loom Video:
https://www.loom.com/share/8237c34fed5c4d82a6c696c844a907ae?sid=b52cc5dc-51c8-4aeb-b87e-b5a90b457cb6

In this task, you will be rating and comparing two responses that were generated from the same coding related prompt by an AI chatbot.

The form is structured in four sections:

1. You will be shown two different responses to the prompt, and asked if you are able to directly attempt to run and test the code blocks present
2. You will be asked to rate the **correctness** of both responses individually.
3. You will be asked to rate how well both responses **follow the instructions** given in the prompt
4. You will then provide a rating on the relative quality of the two responses.

Important notes:

- Questions come with detailed instructions. Please read these carefully. Many questions require explanations depending on the response.
- Some questions require explanations of issues that were found, if any. Aim to write reasonably concise, but not overly generic explanations that would allow a skilled programmer to identify and remedy the problem. Expect to write 1-2 sentences on average.

Out of scope items:
If a prompt is out of scope (e.g. not code related), please select "Cannot Assess" in section 3, and explain why it is out of scope. You can select the other answers arbitrarily; they will be filtered out of the metric calculation.

Skipping items:
You should only rate items where you are familiar with the subject material and very confident that the assessments you're making are accurate. If you are completely unfamiliar with the programming language, unable to assess the correctness of responses, or unable to confidently compare the responses in a reasonable amount of time (~30 minutes), skip the item.

Assessing Correctness:
It is very important to be confident that the ratings you're providing are accurate. Many aspects of the rating task are very subtle and nuanced, where relatively small differences between responses are crucially important, or very reasonable sounding explanations are not factual. Even if at first glance a response seems perfectly appropriate, it is still important to

- Explicitly check each claim and
- Run code blocks yourself (when appropriate and possible) to verify the code runs and works as expected

Rating Task

The primary focus of this rating task is on the **correctness** of the model responses, and how well the responses **follow instructions**. Here, correctness refers both to the truthfulness/factuality of any claim made in the response and the functionality of any code provided. Instruction following refers to how well the response addresses the goals and questions, and requirements of the prompt. It is your job to:
(a) Explicitly research and check these textual claims and
(b) **Run and inspect** the output of any code provided (where applicable) to confirm its executability and functionality.

You will likely need to utilize google search, stack overflow, documentation, or online code executors/compilers (e.g., jsfiddle, colab, programiz, etc.) to confirm that a response is reasonable and that your rating is accurate.

The rating task consists of three parts:

1. Rate the correctness of each of the two responses individually (single-sided).
2. Rate how well each of the two responses follow instructions given in the prompt (single-sided).
3. Rate the relative quality of each response in a side-by-side score.

Task Details:

(1) Code Response Executability:

Are you able to run code provided in the responses?

- Options: Yes-Fully, Yes-Partially, No, No Code Present
- Instructions: Run all code (snippets, functions, programs, etc.) provided in either of the responses, to check both its executability and its correctness.
    - This may not always be possible. For instance, the code may only make sense embedded inside a larger program, or it may require some external file/API dependency for which no execution sandbox is readily available. If the code cannot be executed, select "No".

    - Some responses contain multiple code blocks. If you can run some, but not all of these, answer “Yes-Partially”. Otherwise, answer “Yes-Fully”.
    - *Note*: This question is not asking about the *correctness* of the code, only if there is enough contextual information to test it yourself.
- Explanation: If you answer “Yes-Partially” or “No”, an explanation is required for what code you were not able to run, and why.

(2) Single-Sided Instruction Following:

Did the response follow the instructions it was given in the prompt (both explicit and implicit)?

- Options: No Issues, Minor Issue(s), Major Issue(s), N/A
- Instructions: focus on whether the response reflects the instructions and goals of the prompt, not on truthfulness or correctness issues (e.g., bad code, poor explanation) – those are rated below. Use the following rubric:
    - No Issues: All prompt instructions were followed; response delivered fully on the tasks of the prompt.
    - Minor Issue(s): The response addressed most of the instructions or goal(s) of the prompt, but missed or misinterpreted some small parts. A user would still be reasonably satisfied.
        - Example: a response that describes the right API but assumes a slightly different use-case than what the user articulates.
    - Major Issue(s): Response missed key components of the prompt, rendering it unhelpful to the user.
        - Examples include: a response that discusses a different programming language or library than what the user asked about, or misses a key requirement of the code to be generated.
    - N/A - Not Applicable: There are no explicit or implicit instructions to follow in the prompt or the response is canned (e.g. the model states it cannot do it).
- Explanation: required if issues are found. Describe what aspects of the prompt the response missed or misinterpreted.

(3) Single-Sided Correctness:

Is the response truthful and correct?

- Options: No Issues, Minor Issue(s), Major Issue(s), Cannot Assess, N/A
- Instructions: identify the correctness of any claims in the explanation and whether the code (if any) is correct, executable, functional, and useful. Please take up to 30 minutes to research information across both responses, and explicitly run code snippets as needed and where appropriate. Use the following rubric:

- No Issues: All claims in both the explanation and any code comments are factual and accurate; the code (if any) is functional, safe, and useful.
- Minor Issues(s): either or both of the following are true:
  - Text: primary claims (central to addressing the prompt) are factual / accurate; secondary claims contain meaningful inaccuracies (or unfounded claims).
    - /Examples include: an otherwise correct explanation of a library that uses an incorrect link, or a description of a system that misconstrues a small detail of its design.
  - Code: has minor problems where the main functionality of the code is correct; e.g., it fails to handle an edge case, or is correct but has misleading comments.
- Major Issues(s): either or both of the following are true:
  - Text: primary claims contain meaningful inaccuracies (or unfounded claims), such that the response is not helpful to the user.
    - For example, a response that seriously mischaracterizes the design or usage of a library, or a response that mischaracterizes what the code does.
  - Code: has one or more of the following problems:
    - **Executability:** the program does not compile or run and would require substantial effort to repair.
    - **Functionality:** The code does not, or will not, produce the proper intended output or is broken in a logical/functional fashion.
    - **Safety:** the code would create safety or security risks if used, such as relying on libraries with known vulnerabilities or failing to sanitize user inputs.
      - Do not use this to flag responses that make simplifying assumptions that a user would reasonably be expected to notice and improve, such as using a hard-coded password in a clearly visible location.
    - **Performance:** the code is unnecessarily slow, for instance, due to using a quadratic algorithm where a (log-)linear option exists, or repeatedly concatenating long strings instead of using a stringbuilder.
    - **Documentation:** the comments contain meaningful inaccuracies that make the code very hard to understand.
    - Keep in mind that the code may be functional for the prompter, even if it does not compile or run on your setup. For instance, a response that points to a file only accessible to the prompter, or provides a partial program based on the context provided by the prompter should not be marked as non-functional unless it contains errors that would (likely) manifest in the prompter's programming context.

    - Cannot Assess: Cannot determine validity of claims made in the response. Select this option if properly researching the claims in the response would take >30 minutes.
    - N/A - Not Applicable: No explicit or implicit claims are made in the response and it does not include code. Use for this punts (e.g., “As an AI model I am not capable of responding to this type of question”)
- Explanation: Required if issues are found. Describe all issues. Where possible, categorize code-related issues based on the type of issue (functionality, safety, performance, documentation).

(4) Side-by-Side (SxS) Comparison

SxS Score

- Options: Rate your preference between the two responses on a scale from 1 to 7, where 1 means response A is much better than B, 7 means response B is much better than A, and 4 is neutral.
- Instructions: You should prefer the response that would be more helpful to the user. This is mainly a function of how correct the response is and how well it followed instructions. In general, correctness should be the primary consideration and instruction following secondary, but there may be scenarios where the less correct response is the better one. Use your best judgment.
    - If the two responses are equal in terms of correctness and instruction following, you may want to consider other factors such as verbosity or style:
        - Verbosity: Is the length of the response (both the code and non-code portions) appropriate? The response should include all essential information, while avoiding excessive additional details. Generally, a succinct response should be preferred over a verbose one (all else being equal), but this can depend on preference and context.
        - Style: Does the response use high-quality prose that’s well-organized and easy to read, and whether the included code, if any, is reasonably formatted and includes sufficient and accurate documentation
- Explanation: Always required. Briefly explain the most important considerations in your indicated preference. Relate your motivation to the answers provided above.