Lemur Astrologer Coding
Goal-Oriented Multi-Turn (MT) Coding Reviewer Checklist

Overview:

❗ Changelog

Date:	Change Description
10/31/24	❗ MAJOR update - New sub-category under accuracy: Input Validation ❗ MAJOR update - Rewrites are now required for any issues in the response (including 1 or more minor issues) 1 minor issue is enough to classify a response as “bad”
10/29/24	**Clarification about major issues for presentation - this includes repetition and use of paragraphs for 3+ points** Any scenario where a model response doesn’t follow the stylistic guidelines is considered to be a **major issue**

If a task is in an incorrect category or difficulty, do NOT SBQ it, instead select the correct category in the multiple choice field within the task.

How to Use this Checklist:

This checklist is designed to guide you in reviewing tasks. It gives criteria for assessing the prompt quality and the attempter's ratings.

Make a copy of this template (File > Make a copy) and use it while reviewing

When to FIX vs. SBQ a task:

Terrible prompt?

If the prompt is low quality, spam, or not sufficient for the project guidelines, SBQ the task. ❌

No Significant Failures?

If there are no substantial issues or failures in either model response for a turn, SBQ the task. ❌

Disagree with Ratings?

If you disagree with selected ratings (accuracy, instruction following, etc.) adjust them in the per-turn steps and update the justifications. ✅

Goal or Category Adjustments?

If the goal, category, difficulty or similar elements need updates, make those adjustments. ✅

SxS Rating Correction:

If you believe the **"better" model response remains the same**, but want to adjust the degree (e.g., "slightly better" to "much better"), you may change it without affecting future turns. ✅

If only the **justification needs updating**, please adjust as needed. ✅

Changing which response is "better" requires re-rolling the prompt for future turns to reflect the updated context. 🟠 Move to bullet point 7

Please watch this quick 3 minute video for more information

Rewrite Correction:

For **minor fixes** (grammar, wording, small presentation changes), edit the rewrite without re-rolling future turns. ✅

For example, changing the natural explanation underneath a code block to use bullet points rather than a paragraph.

For **major adjustments** (fixing code bugs, adding missing features), re-roll the prompt to ensure future turns reflect the new context.

If this is the last turn in a task, changing the rewrite will have no affect

For any other turns 🟠Move to bullet point 7.

Major adjustments are changes that will noticeably impact the response in future turns. For instance, if you edit the rewrite to include comments in the code, but the remaining turns are based on a version without comments, this creates a clear inconsistency.

Anytime you need to edit/modify the code in a rewrite, this will be a **major adjustment**

What happens if I need to re-roll the next or current turn?

If you can finish the task (including all re-rolled turns) within a reasonable time, go ahead and make the fix. ✅

If the prompt is too complex or there are too many turns to complete feasibly, SBQ the task. ❌

Prompt Quality

This section helps you evaluate the quality of the prompt. Review each point to make sure that the prompt aligns with the assigned task's requirements. All of these must be true for a correct prompt

1st turn only: Prompt fits the task category:

The prompt is correctly labeled and fits the category (e.g., writing code, fixing bugs).

Does the prompt match the **difficulty level** at the top of the task?
- Yes, then proceed
- No, then edit the difficulty field in the prompt categorization section

Does the subcategory of the prompt meet the requirements?
- Yes, then proceed
- No, then edit the sub-category field - we have to make sure it's correctly labeled

Prompt is clear and detailed**:**

The prompt has all necessary information and isn't confusing.
There are no contradicting requests in the prompt

Correct language and tools used**:**

The prompt uses the right programming language or library as instructed.

For multi-turn **tasks only**, each prompt builds on the previous one:

Each prompt logically continues from the previous one
The conversation is fluid and natural
Does the subcategory of the prompt meet the requirements? It likely won't always be the same and may need to be adjusted.

IMPORTANT: Simplified Example of this common sub-category error:

Task Category: Code Generation/Synthesis
Turn 1 Prompt: “Make a 2d minigolf game using JS”
Turn 1 Category: Text to Code *(makes sense)*
Turn 2 Prompt: “Additionally add a stopwatch to the top right corner”
Turn 2 Category: Text to Code *(❌ bad, it should be **Text to Code Edits**, because we're asking for edits to existing code)*

Ratings Per model response

How to use:

This section guides you through reviewing and evaluating each model response based on the attempter’s rating. For each response, use the criteria below to determine if the rating is accurate.

For ratings of 3 (Major Issue): At least one of the conditions listed under "Major Issue" in the category must be true for a rating of 3.

For ratings of 2 (Minor Issue): At least one of the conditions under "Minor Issue" in the category must be true for a rating of 2.

For ratings of 1 (No Issue): All conditions listed under "No Issue" in the category must be true for a rating of 1.

Instruction Following

If the attempter rated 3 (major issues):

The response fails to fulfill the primary request OR fulfills the primary request but fails to adhere to any constraints.

If the attempter rated 2 (minor issues):

The response fulfills the primary request but does not entirely adhere to all the constraints.
The response could have better handled the ambiguity of the prompt.

If the attempter rated 1 (no issues):

The response fulfills the primary request and all of the constraints.

The response may have incorrect implementation (e.g., mistakes in code) if it shows a deep understanding of the prompt request and constraints in the written text.

The response adeptly handles any potential ambiguity in the prompt.

The response fully adheres to all the prompt's requirements/constraints and doesn't do more than the user requested.

Accuracy of Response (including input validation)

If the attempter rated 3 (major issues):

The code fails to execute due to flawed logic.

The code output does not align with the expected program input.

The response contains false or inaccurate claims.

The code contains severe security vulnerabilities.

Relative to the context of the conversation, the response in the most recent turn does not make sense.

Errors from the prior turn were not corrected and compounded.

No edge cases are covered. The response lacks validation for all inputs, making it vulnerable to errors when faced with unexpected or invalid data inputs.

If the attempter rated 2 (minor issues):

The code executes but contains non-failing minor errors; errors that would show up as a linter error but not a compilation error or would cause a warning upon running the code. An issue such as bad type setting in a dynamic programming language like Python or JS (not TS) would also fall under this category.

The code contains some low-risk security vulnerabilities.

The response is factual and accurate.

Relative to the conversation context, coherence is ambiguous or incomplete.

The response falsely asserts claims that are not fully proven or controversial as fact.

Some edge cases are covered. The response handles most expected inputs but misses certain edge cases, which could lead to potential errors or exceptions under specific conditions.

If the attempter rated 1 (no issues):

The code executes without errors and generates an output that aligns precisely with the model intent.

The code is safe, free of vulnerabilities and follows best practices.

All written text (including inline code comments) is factual and accurate.

Relative to the context of the conversation, the most recent turn makes sense.

Prior turn errors were fully corrected, or there were no prior errors.

All meaningful edge cases are covered. The response includes thorough validation for expected inputs and error handling for invalid data, ensuring robustness and resilience to common input errors.

Optimality and Efficiency

If the attempter rated 3 (major issues):

The model's response contains avoidable inefficiencies—such as poor algorithm design, unnecessary complexity, or incorrect performance.

The code is inefficient, especially where performance matters.

If the attempter rated 2 (minor issues):

The code is relatively performant, but some low-lift optimizations could still be done.

The code mostly adheres to common practices and standards.

The code may not be scalable in real-world large-dataset use cases.

If the attempter rated 1 (no issues):

The code is highly performant and is optimized even for edge cases.

The code is sufficiently performant if the prompt does not request the best performance and if a more efficient option would require much more complexity.

The code adheres to common practices and standards.

Presentation

If the attempter rated 3 (major issues):

Documentation is missing or insufficient to understand the code.

Not Including Any Code Comments

The response has poor readability due to a lack of structure or formatting.

The code is missing programming language tags.

The response includes repetitive content

The response does not replace any paragraphs **with at least 3 points** with bullet points instead

The code has lousy variable or function names.

The logic in the code is disorganized and difficult to follow.

Explanations are absent, leaving the reader unclear about key decisions or steps in the code.

Code is poorly structured, making it challenging to integrate or reuse.

If the attempter rated 2 (minor issues):

The documentation is sufficient to understand the code, but additional details would be helpful.

The response contains a few language mechanics errors, but they do not impact readability.

The response readability can be improved with better formatting.

Some adjustments to formatting and structure could improve clarity, such as adding more bullet points or logical sections.

If the attempter rated 1 (no issues):

The code has an adequate amount of documentation, including in-code comments.

The response is clear, concise, and well-structured.

Variables and functions are named with readability in mind.

All modifications are documented in the code or in a written explanation outside the code.

Out of Date:

If the attempter rated 2 (major issues):

The code uses a deprecated API, library or function or ones with known inefficiencies or vulnerabilities, or are generally not recommended AND stop the code from compiling

If the attempter rated 1 (no issues):

The code uses a maintained library or function that is an older version or less efficient but still allows the code to run.

Rewritten response:

All of these must be true for a correct response rewrite:

General for Reviewing Rewrites:

The rewritten code corrects any errors from previous turns and fully satisfies the prompt's requirements. Making the necessary changes to fix and improve the original model response

Any explanations provided alongside the code are accurate and clarify the changes made or the solution overall.

Code Testing:

If no rewrite is required (in the case of one good and one bad response), does the good response's code run?

Yes, then proceed

No, then change the toggle to say a rewrite is required, and proceed to rewrite the response and proceed (or SBQ)

(When a rewrite is present) Does the code run in the rewrites?

Yes, then proceed

No, then fix the code + re-roll the next prompt or SBQ if you are time constrained

Is the code optimal and if code efficiency (i.e. O(n^2) vs O(1), etc) is described in the response, is it accurately described? Make sure to verify.

Yes, then proceed

No, then update accordingly

Does the code contain sufficient error handling and input validation?

Yes, then proceed

No, then update accordingly

Style and Presentation:

Does the code contain sufficient comments?

Yes, then proceed

No, then add comments + re-roll the next turn or SBQ

Does the rewritten response replace any paragraphs (especially those with at least 3 points) with bullet points instead?

Yes, then proceed

No, then re-word into three bullet points (do not need to re-roll)

Is all repetitive phrasing/wording removed and made more concise?

Yes, then proceed

No, then make the necessary edits

Does the rewritten response satisfy all of the requirements of the prompt?

Act like a lawyer when it comes to the prompt: for every ask or request in the prompt, does the rewritten response satisfy it?

Yes, then proceed

No, then edit the rewrite further to make sure both the text and code address the prompt

When should they do a rewrite?:

For any turn:

The attempter will be required to **always** rewrite the **preferred** response to achieve the goal of the prompt and fix any issues that were identified.

The only scenario where a rewrite would **not** be necessary is if one of the model responses is completely **perfect**, meeting all the requirements and specifications of the prompt. In this case, indicate that a r*esponse rewrite is not needed*. **This is very rare and requires a thorough justification, in most cases, the response will require a rewrite.**
Specification List

The re-written response should **fully achieve the most recent prompt.**
The re-written response should **address any secondary objectives implied by the prompt.**
The re-written response should **correct all errors (major or minor)**
The re-written response should be **coherent and logically connected to the prior conversation.**
The re-written response should **fulfill al the dimension for "No Issues" in the rating rubrics.**
The rewrite should have **input validation** if the code accepts input from the user
The re-written response should **follow the formatting and styling specifications.**
The code of the re-written response should **contain enough comments.**
The code of the re-written response MUST run properly and be optimal and efficient. **Failure to properly test code results in removal from the project.**

In short, they are doing a rewrite step, this response is required to be perfect! All identified issues must be fixed, regardless of their severity.

How to score a task

This rubric is designed to help you accurately score tasks based on the quality of both the prompt and the model response ratings.

In this step, you are grading the quality of the **attempter** who's task you are reviewing.
These are just suggestions to help guide you, if any other issues in the task that are not mentioned below come up, give the task the score that you think it deserves

1. Spam 🚨🚓🚨

The task is irrelevant, nonsensical, or entirely inappropriate. The response doesn't relate to the task category, language, or requirements at all.
The prompt uses code or questions from the internet
Task needs to be SBQd

2. The Prompt or Rewrite is Bad

The prompt is incorrect, misaligned, or does not match the project requirements. The response may not follow key requirements due to the poor prompt setup. In this case the task needs to be redone
The rewrite provides a flawed solution and doesn't meet all of the requirements in the rewrite requirements section
Task needs to be SBQd or requires major fixes

3. 3+ Incorrect Ratings 🟡/ 🟠🔧

There are three or more incorrect ratings for instruction following, accuracy, efficiency, etc. The response contains significant issues in multiple areas that the attempter did not catch, or the attempter flagged issues that were not true.

4. 2 Incorrect Ratings ✅

The task is almost perfect but contains one or two incorrect ratings in key areas (instruction following, accuracy, efficiency, etc.).

5. Perfect Task 🎉🏆

The task is perfect, with a good prompt and all ratings accurately reflecting the response.

Some helpful rubrics:

Rewrite Response Rubric

Field	1-2 (Fail)	3 (Okay)	4-5 (Good/ Perfect)	Additional Notes
Accuracy	• **Major Factual Errors**: Response has 1+ major factual errors or misleading points. • **Minor Factual Errors**: Response has 2+ minor factual errors.	• Contains only 1 minor factual error or misleading statement.	• No factual errors or misleading statements.	• A major error involves incorrect/misleading data central to the request. • A minor error is near the subject matter but doesn't affect the main point.
Instruction Following / Response Fulfillment	• **Main Goal Miss**: Does not achieve the main goal or make progress in the conversation. • **Explicit Instruction Miss**: Misses 1+ key instructions. • **Not Fulfilled**: Fails to answer the question.	• All explicit instructions are followed. • **Secondary Objectives Miss**: Some secondary objectives not addressed. • **Subjective Instruction Miss**: Subjectively misses some parts.	• Fully achieves the main goal or makes clear progress. • Follows all instructions and fully answers the question.	• Rule of thumb: for word count, ±10% is acceptable. • Example: If a question asks for a historical event year, even if implied, the year should be provided.

Unnecessary Greetings / Pleasantries	• Contains greetings/pleasantries such as “Sure, I’d love to help.”	N/A	• No unnecessary greetings or pleasantries.	• Only flag unnecessary phrases at the beginning or end of the response. • Phrases like "Here is..." are not considered pleasantries.
Depth / Nuance	• **Little to No Detail**: Response is superficial and lacks meaningful depth or insight. • **Excessive Detail**: Overly complex and obscures key points.	• Has enough detail but may need more depth or nuance. • **Too Much Detail**: Slightly too much, but doesn’t obscure key points.	• Balanced, insightful, and focused without going overboard.	• Too much detail can lead to confusion. Be clear about what points are most important.
[Rewrite/SxS] Clearly Worse Than Model Response	• **Worse Than Original**: Clearly performs worse than the original across rubric categories. • **Worse Than Side by Side Model**: Performs worse overall.	• Performs similarly to the original or side-by-side comparison.	• Performs better overall than the original or side-by-side comparison.	• Don’t penalize for minimal/no changes if the original response was acceptable. • Compare against state-of-the-art models.