Thank you for your interest in **FriendliAI**.

The following take-home questions are designed to help us better understand you from multiple perspectives, including your technical skills, programming language proficiency, problem-solving style, and intellectual approach.

You **may** use any resources to complete this assignment, including reference materials, external services, or AI tools. If you do so, you are welcome to include a brief addendum with each response describing the materials or AI prompts you used, **only if** you feel this additional context helps illustrate your thinking process.

Each question includes:

- Background related to the problem
- A problem statement
- A suggested time estimate

The time estimates are guidelines rather than strict limits; you are free to spend additional time if needed. If you introduce any assumptions, constraints, or requirements that are not explicitly stated in the prompt, please be sure to clearly note them in your response.

We appreciate the time and thought you put into this assignment and look forward to reviewing your work.

Q1) Python Programming (Suggested time: 15 min)

Background
You are working in a large monorepo and notice that many functions share a common signature: each accepts a dictionary with string keys and whole number values. These functions appear across a variety of contexts, including API services, data processing pipelines, CLI tools, and scheduled billing jobs. To improve the team's developer experience and reduce duplicated validation logic, you decide to introduce a shared utility.

Problem
Your task is to implement a Python 3.11+ decorator that ensures only variables of type `dict[str, int]` are passed to the original function.

Instruction
Share a GitHub Gist or repository link containing your solution. Provide instructions on how to execute the function if necessary. You may have a separate `README.md` file.

Q2) Debugging LLMs (Suggested time: 30 min)

Background
One of the strengths of open-source AI models is that we can customize and fine-tune them to specific needs. However, this flexibility comes with added responsibility: ensuring that all model files and configurations are correctly set up. As a FriendliAI engineer, you want to help users successfully deploy and run open-source models.

Problem - a)
A user provides a Hugging Face repository link and reports that “inference does not work with this model.” Your task is to investigate why the model cannot be used to run a functional `/chat/completions` API server and identify the root cause of the issue.

The link: https://huggingface.co/yunmorning/broken-model

Instructions - a)
Your goal is to review the model’s configuration files and make only the minimal changes required to resolve the problem. Apply your engineering judgment to distinguish between incorrect settings and acceptable variations.

To complete this task:

1. Create a Hugging Face account (if you do not already have one).
2. Create a new, empty Hugging Face repository.
3. Upload the original model files as the initial commit.
4. Apply your fixes and commit the corrected version.
5. In the repository’s README, clearly document which values were changed and explain why those changes were necessary.

Problem - b)
The user reports that including a `reasoning_effort` parameter in requests to this model has no observable effect on the generated output. The model in question is the same model you analyzed and corrected in **Problem (a)**.

Instructions - b)

- Identify the root cause to explain why this parameter is currently ineffective in this context.
- List all necessary steps, requirements, or architectural changes required to make the `reasoning_effort` option meaningful and functional.

Q3) Evaluation Design (Suggested time: 1 hr)

Background
A client is evaluating whether to replace their current open-source inference engine (e.g., vLLM) with the Friendli Engine to improve inference efficiency. To support this decision, they have requested a reproducible benchmark that demonstrates the performance differences between the two systems.

Problem
Your task is to design and implement a benchmarking experiment that compares the inference performance of an open-source engine and the Friendli Engine. The benchmark will be used to quantify efficiency gains and to make comparisons easy to interpret.

Instruction
Create a Python script that the client can run locally or in their own environment to evaluate performance. Your solution should:

- Measure the metrics you believe are most important for demonstrating inference efficiency.
- Execute a fair and reproducible comparison between the two engines.
- Automatically generate **a single graph** that best visualizes the performance gap and supports the conclusion that the Friendli Engine provides superior efficiency.
- Provide a GitHub repository containing: the benchmarking script, instructions on how to run it, and a brief explanation on why you selected the chosen metrics and why the visualization effectively communicates the performance difference.
- You may assume both vLLM and Friendli Engine are already deployed.