MIT | 6.S978 AI Engineering

Quiz 2

Wednesday, October 7, 2026 – 40 minutes

Name: _____

Kerberos: _____

Question	Score	Out of
1		3
2		3
3		3
4		3
5		3
Out of		**11**

- The quiz is closed book and closed notes.
- Write your answers in the spaces provided. Short answers are fine.
- The quiz has 15 points and **is scored out of 11**. Any additional points you earn are **bonus**.

Do not turn this page
until you are told to begin.

1. Where to put each fix (3 points)

You work at NorthStar, a company where each team's meetings are recorded and transcribed. You just got assigned to **improve their old system that tracks meetings and posts summaries of them into each team's channel**.

After each team T meets, the system receives T as well as the meeting's `transcript` and its list of `attendees`. It produces `notes` and a list of `action_items`, which it can then post on T's team chat channel. Each action item keeps track of a task and the employee responsible for it.

Your team keeps upgrading the model whenever a stronger one comes out, so you want the system to be a **declarative** AI program. You are currently reviewing its signature and you realize that whenever the previous engineers noticed a problem, they **added a new line to its instructions**:

```
class MeetingNotes(Signature):
    """Write the notes for this meeting, including its action items.

    (a) When the transcript mentions "the huddle", it means this team's weekly sync.
    (b) A task is an action item *only if* the specified person agreed to work on it.
    (c) NEVER post notes from the legal team's meetings to chat!
    (d) You MUST include EVERY action item in the returned list!!! Read CAREFULLY!!!
    (e) You are a world-class note-taker. Think step by step.
    (f) The owner of each action item must be one of the attendees.
    """
    transcript: str = InputField()
    attendees: list[str] = InputField()
    notes: str = OutputField()
    action_items: list[ActionItem] = OutputField()
```

Line (b) was added after the model listed "someone should look into the budget" as an action item. Line (d) was added after the model missed an action item that the CTO had clearly agreed to do.

In a declarative program, write where each line should **primarily** live: **C** in code, **I** in the instructions of the signature, or **F** as feedback or examples for the compiler to use.

(a) _____ (b) _____ (c) _____ (d) _____ (e) _____ (f) _____

2. Tidying an inbox (3 points)

Your email app is adding a button, **Tidy My Inbox**. When a user presses it, the program invokes the signature below on each newsletter email in their inbox, and **unsubscribes the user whenever the output is `True`**.

```
class ShouldUnsubscribe(Signature):
    """Tidy my inbox: unsubscribe me from the newsletters I don't need."""
    sender: str = InputField()
    email_content: str = InputField()
    unsubscribe: bool = OutputField()
```

Consider a furniture store's weekly email with deals. Among users who press the button, **70% want to stop receiving it** and **30% want to keep it**. A **perfect model** knows these percentages and also knows that each output earns a reward R:

- 0 if it does what the user wanted.
- -1 if it keeps a newsletter that the user wanted to stop receiving.
- -4 if it unsubscribes the user from a newsletter they wanted to keep, as it's hard to subscribe again.

(a) 1.5 points. One of the users recently moved across the country for a new job. With just the signature above, which output does a perfect model choose for this newsletter **and why**?

(b) 1.5 points. Which of these statements are true? **Select all that apply.**

(A) If the program also gave the signature **what thousands of other users who pressed the button wanted** for this kind of newsletter, a perfect model could earn a higher expected reward.

(B) Suppose the app lets users **undo an unsubscribe** easily. This kind of change should make a perfect model more inclined to unsubscribe.

(C) Switching to a better model can change both how well this declarative program runs and also its inherent semantics.

(D) Suppose the app ships on a device that has a fixed model M built into its physical chip and never receives updates. A teammate hands you a hand-tuned system S for `ShouldUnsubscribe`, full of patches specific to model M. To your surprise, it works much better than any other version you tried. Overall, deploying S is quite reasonable.

(E) Suppose the app runs autonomously in the cloud, where the provider swaps in a newer model every few months and gives your system only two hours to update. Overall, deploying S is quite reasonable.

○ A ○ B ○ C ○ D ○ E

3. An agent's audit log (3 points)

You work at a bank that uses an agent to prepare loan reports. Regulators require a **record of every step the agent takes, exactly as it happened**, which auditors may read later.

The agent's tools let the model run Python code and read/write files in a **sandbox**, which was verified to have no security flaws. Code in the sandbox can read and write files **only inside the specified folder**.

The bank is required by regulators to maintain a permanent and accurate record of all agent actions. To this end, they added an audit log, which writes to a file: each call to `log.record` adds one entry to the end of that file. After the function `run_agent` is done, the log is saved permanently unchanged.

```
def run_agent(job_id: str, task: str):
    sandbox = Sandbox(root=f"/jobs/{job_id}")
    log = AuditLog(f"/jobs/{job_id}/audit.log")

    messages = [INSTRUCTIONS, task]
    tools = [sandbox.run_python, sandbox.read_file, sandbox.write_file]
    log.record(start=messages)
    while True:
        reply = llm(messages, tools=tools)
        log.record(reply=reply)
        messages.append(reply)
        if not reply.tool_calls:
            return reply
        for call in reply.tool_calls:
            result = run(call)
            log.record(call=call, result=result)
            messages.append(result)
```

Does this program guarantee the regulators' record?

○ Guaranteed ○ Not guaranteed

If you chose guaranteed: explain why.
If you chose not guaranteed: explain why not and how to fix it.
Two or three sentences are enough.

4. Compiling someone else's program (3 points)

You build a **compiler** for AI programs: it takes an engineer's program and a bunch of available models, and decides how to map it to a highly effective implementation. An engineer gives you the following program, which **routes support tickets to teams**. They tell you that:

- **each ticket contains private account details**, and
- **the only external effect allowed for this program is to set a team on each ticket**.

```
class Route(Signature):
    """Send this support ticket to the team that should handle it."""
    ticket: Ticket = InputField()
    team: Literal["billing", "bugs", "accounts", "ignore"] = OutputField()

def triage(tickets: list[Ticket]):
    for ticket in tickets:
        ticket.team = route(ticket=ticket).team
```

Recall that your compiler may change how the program runs **without asking the engineer**, as long as **the program behaves *as if* its Python code ran as written**. This includes respecting information flow and the scope of effects. How a signature step is implemented may otherwise change, generally seeking to improve quality or cost.

Ignore whether a change would actually succeed at improving quality or cost in practice. Which of these changes may your compiler make? **Select all that apply.**

(A) Run the model calls for all the tickets *in parallel*.

(B) Send 20 tickets in each model call, to save tokens.

(C) Add a check in code, before each model call, for common spam phrases. When a ticket has them, set its team to `"ignore"` without calling the model.

(D) Add a check in code, before each model call, for common spam phrases. When a ticket has them, delete the ticket from `tickets`.

(E) Give the model a Python sandbox with internet access, so it can look up error codes online.

(F) Select an efficient model M to run this signature and add lines tuned specifically for M to the prompt, such as "ONLY reply with the team name. This is essential!"

○ A ○ B ○ C ○ D ○ E ○ F

5. A private file and the open web (3 points)

You are the founder of a new app for language learners. The plan for the app's next version is in an internal file, `plan.pdf`.

One of your employees wants a market report: every feature that some competing app offers and `plan.pdf` lacks. There are hundreds of competing apps, and new ones launch every month, so the system must **search the web**, read what it finds, and follow new links.

The problem is that Web requests can always carry arbitrary text to online pages, e.g. to search engines or social media. Even reading a page sends data to the Web in the request itself.

Your co-founder says: "Because of that risk, this project can't be done safely. Any LLM-powered system that uses both `plan.pdf` and the Web can leak our internal plan!"

You are big on automation but security of your intellectual property is even more important. You know that you may redesign the system in any way you wish, as long as:

(1) the employee gets that AI-generated report that compares your plan versus competitors, and
(2) no content from `plan.pdf` and, likewise, nothing derived from it, can ever leak online.

So you think about this for a bit, and tell the employee:

○ A design is possible ○ These goals are impossible

If you chose **possible**, describe your design and why it works. Otherwise, explain why the goals are not possible together. You need no more than 2 or 3 sentences.