

## PRACTICE QUESTIONS

Microsoft

# GH-600

GitHub Certified: Agentic AI Developer

SHARED BY [u/lively\\_pencil\\_26](#) on aiexamcerts

|           |                                                          |
|-----------|----------------------------------------------------------|
| EDITION   | Free Set 1                                               |
| QUESTIONS | 10                                                       |
| SHARED    | October 18, 2026                                         |
| EXAM      | Microsoft GH-600, GitHub Certified: Agentic AI Developer |

Shared by [u/lively\\_pencil\\_26](#) on aiexamcerts

10 practice questions for the Microsoft GH-600 exam, GitHub Certified: Agentic AI Developer, each followed by its answer and a short explanation. More free sets and the full practice exam: [aiexamcerts.com/microsoft/gh-600](https://aiexamcerts.com/microsoft/gh-600). Spotted a mistake? Email [support@aiexamcerts.com](mailto:support@aiexamcerts.com) with the exam code GH-600 and the question number.

---

1. An agent must recall a team's coding conventions across many unrelated tasks, but the temporary notes from one debugging session should be dropped when that task ends. Which pairing matches this best?
- A Short-term context for the conventions and an external store for the debugging notes
  - B Short-term context for both, since both are needed only once
  - C Long-term memory for the debugging notes and no memory for the conventions
  - D Long-term memory for the conventions and short-term context for the debugging notes

**Answer: D**

Durable conventions that apply across tasks belong in long-term memory, while task-specific notes belong in short-term context that is cleared at the end. Option A is tempting because it puts the conventions in the wrong category, which would cause them to vanish between tasks. Keeping both in short-term context loses the conventions, and keeping no memory for them repeats the same mistake.

---

2. Agent progress that is kept only in the model's context window will still be available after the session restarts, so no durable artifact is needed.
- A True
  - B False

**Answer: B (False)**

A context window is cleared when a session ends, so progress held only there is lost on restart. Durable artifacts such as a progress file, a plan document, or the commit history let the agent resume without repeating steps. The statement is tempting because the agent seems to remember earlier turns during a session.

---

3. An agent resumes a multi-day migration and repeats a step it already completed. What is the most likely missing practice?
- A Recording completed steps and decisions in a durable progress artifact that the agent reads at the start of each run
  - B Increasing the context window so earlier messages fit in one prompt
  - C Running the agent with a higher temperature so it explores new paths
  - D Deleting the migration branch before each resume, so that the agent starts from a clean slate and no older partial state can interfere with its new work

**Answer: A**

Capturing progress and decisions in a durable artifact that is read at resume time lets the agent continue without repeating completed work. A bigger context window is tempting because it seems to solve forgetting, but it does not survive across runs. Temperature and deleting the branch do not address recorded progress at all.

4. Two parallel agents record conflicting assumptions about an API shape in their own notes. Which practice prevents this kind of conflicting context?
- A Giving both agents the same random seed so their notes match word for word, which keeps their outputs identical across every run and every session
  - B Keeping one shared source of truth for decisions that both agents read before they act
  - C Letting each agent overwrite the other's memory on a fixed timer
  - D Disabling memory so each agent starts from a blank state every time

**Answer: B**

A single shared source of truth for decisions gives both agents the same facts before they act, which is the direct fix for conflicting context. Matching seeds is tempting because it sounds like synchronization, but seeds control randomness, not recorded decisions. Blank states remove the conflict only by removing the shared history that the team needs.

5. To stop an agent from acting on outdated information, the team sets a memory \_\_\_\_\_ rule so that entries older than a chosen period are removed.

**Answer: expiration** (also accepted: expiry, expiration rule, expiry rule)

An expiration rule removes memory entries after a set period so the agent does not act on stale facts. Pruning is related but removes entries that are no longer relevant, not entries that have simply aged out. The accepted answer is expiration, with expiry as an accepted variant.

6. Which signal is the best quantitative evaluation signal for an agent's pull request?
- A A reviewer's note that the code feels clean and well organized, recorded in the pull request thread by whoever reviewed it most recently
  - B The agent's own statement that the task was completed successfully
  - C The number of failing tests or security findings reported by automated scans on its branch
  - D Whether the pull request description is written in the first person

**Answer: C**

A count of failing tests or scan findings is a measurable signal that can be compared across runs. A reviewer's impression is a qualitative signal, which is tempting because reviewers do judge quality, but it is not a count. The agent's own claim of success is not independent evidence, and the writing person of the description has no bearing on correctness.

7. An agent called the wrong tool with the correct arguments because the tool's description was ambiguous. How should this failure be classified?
- A Context drift during a long session
  - B An environment constraint that blocked the intended action
  - C A reasoning error about the overall goal of the task
  - D Tool misuse

**Answer: D**

Calling the wrong tool is tool misuse, and an ambiguous description is one of the causes to fix when tuning tool access. Context drift is tempting because the agent did lose track of the right tool, but the failure happened at the moment of the call. A reasoning error about the goal would mean the plan itself was wrong, which it was not here.

8. Which two practices reduce the risk of overlapping code changes when several agents work in one repository at the same time? (Choose two.)
- A Give each agent its own branch or worktree so that its edits are isolated
  - B Let all agents push directly to the same branch to save time on merges
  - C Assign each agent a separate set of files or modules, and sequence the tasks that must touch the same code
  - D Merge every agent's output without review to keep the queue moving
  - E Disable pull request conflict detection so that no agent is blocked

**Answer: A, C**

Isolated branches or worktrees keep each agent's edits separate until they are reviewed, and disjoint file assignments prevent collisions. Shared branches look efficient, yet they guarantee overlapping edits. Merging without review and disabling conflict detection both remove the checks that would catch the overlap.

9. Revising the repository's \_\_\_\_\_ file is a common way to tune agent behavior across every run that uses the repository.

**Answer: custom instructions** (also accepted: copilot-instructions.md, instructions)

Repository-wide custom instructions apply to every agent run in that repository, so a fix written there tunes behavior everywhere at once. Editing a single agent profile is narrower and is tempting when only one agent misbehaves. The accepted answers are custom instructions, with `copilot-instructions.md` and `instructions` as variants.

10. A coordinator agent that delegates subtasks to specialist agents should record each handoff and its outcome so that a failed subtask can be traced after the run.

- A True
- B False

**Answer: A** (True)

A recorded handoff with its input and outcome lets reviewers see which specialist produced a bad result and why. Without that record, a failure in one subtask is hard to separate from the rest of the run. The statement is true because traceability is what post-hoc analysis depends on.