Multiple-Choice Questions on Multi-Agent Orchestration

**1. What does orchestration refer to in the context of the Agents SDK?**

A. Model tuning strategy  
B. Workflow flow: which agents run, order, and decision logic  
C. Data preprocessing pipelines  
D. Memory management  
**Answer:** B — Orchestration determines which agents run, in what order, and how decisions are made

**2. What are the two main orchestration patterns described?**

A. Centralized and decentralized databases  
B. ML-based and rule-based pipelines  
C. LLM-driven orchestration and code-based orchestration  
D. Streaming and batch execution  
**Answer:** C — You can orchestrate via the LLM’s own planning or explicitly via code logic

**3. In LLM-driven orchestration, what capabilities are agents expected to have?**

A. Static prompts only  
B. Only local context access  
C. Planning, tool use, and agent handoffs  
D. External APIs only  
**Answer:** C — Agents can autonomously plan, use tools, and hand off tasks to sub-agents

**4. Which of the following is *not* one of the examples of tools agents may use in orchestration?**

A. WebSearchTool  
B. FileSearchTool  
C. LocalShellTool  
D. Experimental AI Debugger  
**Answer:** D — The examples include Web search, file search, computer use, code execution, and handoffs—but not an "Experimental AI Debugger"

**5. What is the code-based orchestration pattern?**

A. Training agents via code signals  
B. Defining agent flow and sequencing manually using code  
C. Automatically generating code from prompts  
D. Orchestrating via configuration files  
**Answer:** B — You define in code which agent runs when and how control passes

**6. Can you combine both orchestration styles?**

A. No, you must choose one  
B. Yes — mix LLM-driven and code orchestration as needed  
C. Only via special APIs  
D. Only in JS SDK  
**Answer:** B — Mixing both orchestration styles is supported and common

**7. What metaphor is used to describe a central planner pattern in multi-agent workflows?**

A. Handoff tree  
B. Manager-agent with tool-call edges  
C. Graph database  
D. Circular pipeline  
**Answer:** B — In the “manager” pattern, a central agent calls specialist agents as tools, represented by edges in a graph

**8. What is the primary benefit of the manager pattern?**

A. Enables peer-to-peer communication  
B. Centralizes orchestration and preserves user context  
C. Reduces model size  
D. Optimizes tool usage cost  
**Answer:** B — A manager agent orchestrates tasks, delegates to specialists, and synthesizes results with context preservation

**9. In contrast to the manager pattern, what describes a decentralized multi-agent approach?**

A. One agent controlling everything  
B. Multiple specialist agents handing off control among each other  
C. Only using tools, no agents  
D. Single execution thread  
**Answer:** B — Decentralized means agents hand off tasks to other agents, each owning part of the problem

**10. In the portfolio manager example, what role does the head agent fill?**

A. Final answer formatter  
B. Central planner that invokes specialized agents  
C. Logging coordinator only  
D. Context memory manager  
**Answer:** B — The portfolio manager orchestrates specialized macro, fundamental, and quantitative agents

**11. Which orchestration pattern is described in the portfolio collaboration example?**

A. Nested loops only  
B. Agent-as-tool approach with central manager agent  
C. Decentralized handoffs without a leader  
D. Rule-based fallback system  
**Answer:** B — It uses the “agent-as-tool” style with a central manager invoking specialized agents

**12. Why is the agent-as-tool pattern beneficial?**

A. Increases LLM compute demand  
B. Facilitates auditable, scalable workflows with single-point orchestration  
C. Removes need for memory  
D. Reduces need for tools  
**Answer:** B — It provides modularity, transparency, and centralized control via tool-like agent calls

**13. Is observability supported in multi-agent orchestration?**

A. No — tracing only works for single agents  
B. Yes — through Tracing and tools like Dynatrace  
C. Only via manual logs  
D. Only in JS SDK  
**Answer:** B — Built-in tracing supports visibility into multi-agent workflows, and external observability tools like Dynatrace can also be integrated

**14. Which two orchestration modes are adopted by the Azure Foundry multi-agent demo?**

A. Code workflow plus manager tool call  
B. LLM-driven + code orchestration  
C. Streaming only  
D. Event-driven only  
**Answer:** B — It combines LLM-decision and code-based orchestration for workflows like research, summarization, and translation

**15. In the Dynatrace example, what does the Welcome Agent do?**

A. Final response formatting  
B. Greet user, analyze prompt intent, route to the right agent  
C. Summarize search results  
D. Translate content  
**Answer:** B — The Welcome Agent handles engagement, intent analysis, and routing in the agent network

**16. Which specialist agents are used in the Azure-based multi-agent system?**

A. Editor, publisher, validator  
B. Researcher, summarizer, translator  
C. Logger, tracker, orchestrator  
D. Sentiment analyzer, planner, executor  
**Answer:** B — There are researcher, summarizer, and translator agents alongside the welcome/planner agent

**17. What role does handoffs play in multi-agent orchestration?**

A. A tool for switching agent control within conversations  
B. A method of reducing token usage  
C. An error-handling mechanism  
D. A debugging feature only  
**Answer:** A — Handoffs act as a mechanism for agents to delegate and pass control to others mid-execution

**18. Can orchestration include parallel agent execution?**

A. No — only sequential flow supported  
B. Yes — e.g., parallel agent patterns in examples  
C. Only with streaming mode  
D. Only external orchestration supports it  
**Answer:** B — Parallel agent execution is shown in examples and supported in the SDK

**19. What design principles are emphasized for multi-agent orchestration?**

A. Monolithic blocks  
B. Flexibility, composability, clear prompts  
C. Minimal tracing logs  
D. Agent merging only  
**Answer:** B — Orchestration should emphasize flexible, composable designs with clear prompt structure

**20. Which orchestration pattern maintains a single control thread while delegating tasks?**

A. Decentralized handoff pattern  
B. Manager (agent-as-tool) pattern  
C. Streaming events pattern  
D. Guardrails pattern  
**Answer:** B — The manager or agent-as-tool pattern keeps a unified orchestration flow while invoking specialists.