Tutorial 6：推理系统架构

题目

1. [单选题] 下列哪组对象通常不能从“最小推理系统图”中省略？A. 请求入口、主控、调度、执行、状态层、结果返回 B. 仅保留 README 与日志目录 C. 只保留模型权重文件 D. 只保留 API 和 GPU 名称

---

2. [判断题] 目录图可以直接等同于架构图，因为它们都展示了模块名称。

---

3. [多选题] 下列哪些对象更接近控制层？A. engine B. scheduler C. model runner D. 纯数据文件

---

4. [简答题] 为什么状态管理层值得在架构图中被单独画出？

5. [匹配题] 请将下列对象与职责匹配：
   - engine
   - scheduler
   - model runner
   - block/state manager 职责选项：请求主控、执行顺序分配、模型执行推进、状态组织与分配。

---

6. [判断题] 只有做重计算的模块，才可能成为系统关键控制点。

---

7. [简答题] 为什么 API 层与控制层不能被视为同一回事？

8. [伪代码题] 请写出一个简短伪代码，按 `request -> engine -> scheduler -> runner -> return` 的顺序输出最小控制链。

9. [多选题] 当系统走向分布式时，哪些边界通常会首先发生变化？A. 控制边界 B. 状态边界 C. 资源放置边界 D. 课程标题页样式

---

10. [简答题] 什么样的图才足以支撑系统级讨论？

参考答案

1. A。这些对象通常都不能省略。
2. 错。目录只说明摆放方式，不说明职责和边界。
3. A、B。engine 和 scheduler 更接近控制层。
4. 因为状态对象会反向约束调度、并发和资源利用。
5. engine -> 请求主控；scheduler -> 执行顺序分配；model runner -> 模型执行推进；block/state manager -> 状态组织与分配。
6. 错。控制错误本身就足以拖垮整体系统行为。
7. 因为 API 层负责接入，而控制层负责决定系统行为。
8. 示例伪代码：

```
function minimal_path(request):
    enter(engine)
    enter(scheduler)
    enter(runner)
    return result
```

9. A、B、C。这些边界在分布式场景下通常都会先变化。
10. 至少应能说明对象、边界、控制关系和关键路径。