实验七：人体快速排序计算机 - 详细执行计划

一、 实验核心目标

1. **算法理解**: 通过物理模拟，深刻理解快速排序（Quicksort）的分治思想和串行执行

   3. 正确性现场验证

   为确保实验的严谨性和准确性，在排序完成后，需要按以下步骤验证三个层面的正确性：

   a) **结果正确性验证**：
      - 控制器带领全队从左到右依次进行身高比较
      - 确认每一对相邻学生的身高关系都是"左高右低"
      - 如发现任何不符合规则的排序，立即进行修正

   b) **算法正确性验证**：
      - 监督器复查记录的全部操作过程
      - 确认每次分区操作都正确选择了基准
      - 验证每个子区间的划分都符合快速排序的分治逻辑
      - 核实所有比较和交换操作都遵循了算法规则

   c) **系统正确性（串行执行）验证**：
      - 监控器展示拍摄的关键节点照片
      - 确认每一步操作都是由控制器发出的明确指令
      - 核实数据组成员没有进行过任何自发的移动或交换
      - 查看是否存在任何并行操作的情况

2. **详细执行示例**

   为了让流程更具体，我们假设有6名数据组成员，分别是同学A, B, C, D, E, F，他们的身高分别为：`[175, 185, 170, 180, 178, 190]`。初始队列按学号排列。我们的目标是按身高**从高到低**排序。

   初始状态:

   `[ A(175), B(185), C(170), D(180), E(178), F(190) ]`

   (监控器拍照：记录初始状态)

   系统思维: 将小组构建为一台“人体计算机”，理解控制器、运算器、监督器等基本组件的协同工作模式。

3. **正确性证明**: 在实验全程中，确保并能证明以下三点：
      - **结果正确性**: 最终学生的排列顺序严格按照身高从高到低。

- **算法正确性**: 整个过程严格遵循了快速排序算法的每一步逻辑。
- **系统正确性**: 所有操作均按“指令”串行执行，无并行或违规操作。

二、 实验准备阶段 (关键阶段)

1. 团队组建与角色分配 (每组26人)

- **控制组 (4人)**: 整个“人体计算机”的大脑。
    - **监督器 (1人)**: **[算法正确性负责人]** 手持快速排序伪代码或流程图，监督每一步操作是否合规。出现错误时，立即喊“停”，并指导回溯到上一个正确状态。
    - **控制器 (1人)**: **[系统正确性负责人]** 团队的唯一指令发出者。负责选择基准（Pivot）、指挥学生进行身高比较和位置交换。所有数据组同学只听从控制器的指令。
    - **监控器 (1人)**: **[结果记录负责人]** 使用手机或相机，负责拍摄关键步骤的照片和录像，作为后期分析和汇报的材料。重点记录初始状态、每次分区后的状态和最终状态。
    - **计数器 (1人)**: **[性能分析负责人]** 准备纸笔或计数器，每执行一条核心指令（如“比较”、“交换”）就计数一次，最终统计总步数。
- **数据组 (22人)**: 担当“数据元素”和“运算器”。
    - 初始状态：按学号从小到大站成一排。
    - 执行任务：听从“控制器”的指令，进行身高“比较”和位置“交换”。

2. 物资与场地准备

- **场地**: 室外空地（如操场），确保有足够空间进行队列移动。
- **标识物 (可选)**: 可以使用粉笔或标志物在地面画出“基准区”、“小于基准区”、“大于基准区”，使分区过程更清晰。
- **记录工具**:
    - **监督器**: 打印或手写一份清晰的快速排序算法伪代码/流程图。
    - **计数器**: 纸、笔或一个物理计数器。
    - **监控器**: 充满电的手机或相机。
- **身份标识 (可选)**: 为控制组的4名同学制作简单的身份标签，强化角色意识。

3. 制定“指令集” (保障系统正确性的核心)

“控制器”只能使用以下有限的指令来指挥“数据组”，确保操作的原子性和串行性。

- `SELECT_PIVOT(index)`: 选择指定索引的同学作为本次分区的基准。
- `COMPARE(index_A, index_B)`: 命令索引A和索引B的两位同学比较身高。
- `SWAP(index_A, index_B)`: 命令索引A和索引B的两位同学交换位置。
- `MOVE_TO(index, position)`: 命令指定索引的同学移动到某个逻辑位置（如“小于区”末尾）。
- `PARTITION(start_index, end_index)`: 宣布对某个范围的子队列开始进行分区操作。
- `HALT`: 暂停或结束。

4. 明确正确性保障措施

- **保障结果正确**: 排序结束后，由“控制器”从头到尾依次检查每位相邻的同学，确认身高是否为严格递减。
- **保障算法正确**: “监督器”是关键。在实验前，监督器必须完全理解快速排序的每一个细节。实验中，每当“控制器”发出指令，“监督器”需在自己的流程图上同步标记，确保无遗漏、无错序。
- **保障系统正确**: “控制器”是唯一指令出口。所有“数据组”成员被告知，在实验期间不得自行移动或交流，只能被动地执行来自“控制器”的指令。
- **寄存器（可选）**：为了把“寄存器”这一概念可视化，现场可临时指定1–2名同学担任“寄存器”角色，负责记录并展示当前的基准（pivot）、分区边界 i/j 等临时变量。
    - 目的：作为短期存储的可视化载体，帮助监督器和监控器核对每一步的临时变量变化。
    - 要求：寄存器仅在被控制器或监督器明确要求时更新，不得主动更改展示内容，以保持系统串行性。

三、 实验执行阶段

1. **准备阶段 (5分钟)**
    - 数据组同学按学号排好队。
    - 控制组各成员到位，向数据组明确指令和规则。
    - 监控器拍下初始队列（输入状态）。
2. **执行阶段 (核心)**

- **控制器**发出 `PARTITION` 指令，宣布对整个队列进行操作。
- **控制器**发出 `SELECT_PIVOT` 指令，选定一名同学为基准（例如，可以选择队列的第一个人、最后一个人，或随机选择以优化性能）。
- **控制器**使用 `COMPARE` 和 `SWAP` (或 `MOVE_TO`) 指令，将队列中的其他同学与基准比较，并移动到正确的分区。
- **计数器**在每次 `COMPARE` 和 `SWAP` 时进行计数。
- **监督器**全程紧盯，确保分区逻辑无误。
- 一次分区结束后，基准同学的位置被固定。**监控器**上前拍照记录当前状态。
- **控制器**对基准左侧和右侧的两个子队列，递归地重复以上分区过程。
- 直到所有子队列都不可再分（只含一个或零个元素），排序结束。
- **监控器**拍下最终队列（输出状态），并录制一段简短的视频用于展示。

3. **现场正确性验证清单**

   为确保实验的严谨性和准确性，在排序完成后，需要按以下步骤验证三个层面的正确性：

   a) **结果正确性验证**：
   - 控制器带领全队从左到右依次进行身高比较
   - 确认每一对相邻学生的身高关系都是"左高右低"
   - 如发现任何不符合规则的排序，立即进行修正

   b) **算法正确性验证**：
   - 监督器复查记录的全部操作过程
   - 确认每次分区操作都正确选择了基准
   - 验证每个子区间的划分都符合快速排序的分治逻辑
   - 核实所有比较和交换操作都遵循了算法规则

   c) **系统正确性（串行执行）验证**：
   - 监控器展示拍摄的关键节点照片
   - 确认每一步操作都是由控制器发出的明确指令
   - 核实数据组成员没有进行过任何自发的移动或交换
   - 查看是否存在任何并行操作的情况

4. **详细执行示例 (以6名数据组成员为例)**

   为了让流程更具体，我们假设有6名数据组成员，分别是同学A, B, C, D, E, F，他们的身高分别为：`[175, 185, 170, 180, 178, 190]`。初始队列按学号排列。我们的目标是按身高**从高到低**排序。

   初始状态:

   `[ A(175), B(185), C(170), D(180), E(178), F(190) ]`

(监控器拍照：记录初始状态)

第一轮分区：对整个队列 `[0...5]` 操作

i. **控制器**: “开始对索引0到5的队列进行分区！`PARTITION(0, 5)`”

ii. **控制器**: “选择索引5的同学F(190)为基准！`SELECT_PIVOT(5)`”

- *基准值 Pivot = 190。*
- *“高于基准”区域的边界 `i` 初始化为 -1。*

iii. **控制器**: “开始遍历！索引 `j` 从0到4。”

- **j=0**: “比较索引0的同学A(175)和基准F(190)！`COMPARE(0, 5)`”
  - 结果：A(175) < F(190)，A同学身高**不高于**基准，原地不动。
- **j=1**: “比较索引1的同学B(185)和基准F(190)！`COMPARE(1, 5)`”
  - 结果：B(185) < F(190)，B同学身高**不高于**基准，原地不动。
- **j=2**: “比较索引2的同学C(170)和基准F(190)！`COMPARE(2, 5)`”
  - 结果：C(170) < F(190)，C同学身高**不高于**基准，原地不动。
- **j=3**: “比较索引3的同学D(180)和基准F(190)！`COMPARE(3, 5)`”
  - 结果：D(180) < F(190)，D同学身高**不高于**基准，原地不动。
- **j=4**: “比较索引4的同学E(178)和基准F(190)！`COMPARE(4, 5)`”
  - 结果：E(178) < F(190)，E同学身高**不高于**基准，原地不动。

iv. **控制器**: “遍历结束！‘高于基准’区域的边界 `i` 仍为-1。现在将基准归位！”

v. **控制器**: “交换索引 `i+1` (即0) 和基准索引5！`SWAP(0, 5)`”

- 同学A(175)和同学F(190)交换位置。

第一轮分区后状态:

`[ F(190), B(185), C(170), D(180), E(178), A(175) ]`

- 基准 F(190) 已在正确位置（索引0）。
  (监控器拍照：记录第一次分区后的状态)

第二轮分区：对基准右侧的子队列 `[1...5]` 操作

- 此时，我们得到一个新的、规模更小的排序问题：对队列 `[ B(185), C(170), D(180), E(178), A(175) ]` 进行排序。

i. **控制器**: “开始对索引1到5的队列进行分区！`PARTITION(1, 5)`”

ii. **控制器**: “选择索引5的同学A(175)为基准！`SELECT_PIVOT(5)`”
   - *基准值 Pivot = 175。*
   - *“高于基准”区域的边界 `i` 初始化为 0 (low-1)。*

iii. **控制器**: “开始遍历！索引 `j` 从1到4。”
   - **j=1**: “比较索引1的同学B(185)和基准A(175)！`COMPARE(1, 5)`”
     - 结果：B(185) > A(175)，B同学**高于**基准。
     - **控制器**: “边界 `i` 增加1，变为1。交换索引 `i`(1) 和 `j`(1)！`SWAP(1, 1)`” (B同学原地不动)
   - **j=2**: “比较索引2的同学C(170)和基准A(175)！`COMPARE(2, 5)`”
     - 结果：C(170) < A(175)，C同学**不高于**基准，原地不动。
   - **j=3**: “比较索引3的同学D(180)和基准A(175)！`COMPARE(3, 5)`”
     - 结果：D(180) > A(175)，D同学**高于**基准。
     - **控制器**: “边界 `i` 增加1，变为2。交换索引 `i`(2) 和 `j`(3)！`SWAP(2, 3)`” (C和D交换位置)
     - 当前队列: `[ F(190), B(185), D(180), C(170), E(178), A(175)`
   - **j=4**: “比较索引4的同学E(178)和基准A(175)！`COMPARE(4, 5)`”
     - 结果：E(178) > A(175)，E同学**高于**基准。
     - **控制器**: “边界 `i` 增加1，变为3。交换索引

`i`(3) 和 `j`(4)！`SWAP(3, 4)`” (C和E交换位置)

- 当前队列:
  `[ F(190), B(185), D(180), E(178), C(170), A(175) ]`

iv. **控制器**: “遍历结束！‘高于基准’区域的边界 `i` 为3。现在将基准归位！”

v. **控制器**: “交换索引 `i+1` (即4) 和基准索引5！`SWAP(4, 5)`”

- 同学C(170)和同学A(175)交换位置。

第二轮分区后状态:

`[ F(190), B(185), D(180), E(178), A(175), C(170) ]`

- 基准 A(175) 已在正确位置（索引4）。
  (监控器拍照：记录第二次分区后的状态)

后续递归过程

- 现在，队列被进一步分割成 `[1...3]` 和 `[5...5]` 两个子问题。
- 对 `[5...5]` (只含C(170)) 的排序，因 `low < high` 不成立而直接结束。
- 对 `[1...3]` (即 `[ B(185), D(180), E(178) ]`) 的排序会继续递归下去，直到所有同学都归位。
- ... 这个过程不断重复，直到所有子队列都只有一个或零个元素。

最终状态:

`[ F(190), B(185), D(180), E(178), A(175), C(170) ]`

(监控器拍照：记录最终排序完成的状态)

排序完成后的正确性验证

在这个六人示例中，我们按照以下步骤验证三个层面的正确性：

i. **结果正确性验证**

- 控制器带领全队从F到C依次检查：
  - F(190) > B(185) ✓
  - B(185) > D(180) ✓
  - D(180) > E(178) ✓
  - E(178) > A(175) ✓
  - A(175) > C(170) ✓

- 确认每一对相邻同学都满足"左高右低"的顺序要求

ii. **算法正确性验证**

- 监督器检查整个执行过程的记录：
    - 第一轮：正确选择F为基准，完成分区
    - 第二轮：正确选择A为基准，完成分区
    - 子队列处理：符合快速排序的递归分治逻辑
- 确认所有分区操作都正确执行了Lomuto分区方案

iii. **系统正确性（串行执行）验证**

- 监控器展示三张关键照片：
    - 初始状态：`[A(175), B(185), C(170), D(180), E(178), F(190)]`
    - 第一轮后：`[F(190), B(185), C(170), D(180), E(178), A(175)]`
    - 最终状态：`[F(190), B(185), D(180), E(178), A(175), C(170)]`
- 确认整个过程中：
    - 所有移动都由控制器指令驱动
    - 同学们没有自行移动或交换
    - 没有发生并行操作

注：为了把“寄存器”这一概念可视化，现场可临时指定1名“寄存器”负责记录并展示当前的基准（pivot）与分区边界（i/j），由监督器在执行过程中按需指派和核对；监控器可把寄存器展示的内容作为拍照记录的一部分，便于事后验证临时变量的变化。

通过这个生动的例子，每个组员都能直观地看到数据是如何在指令的驱动下，一步步从无序走向有序的，同时也展示了如何严格验证实验的正确性。

四、 课后阶段：总结与汇报

1. **材料汇总 (各组)**
    - **监控器**整理照片和视频，挑选出能清晰展示算法流程的素材。
    - **计数器**报告最终的指令总步数。
    - 全体组员共同讨论“思考题”，并记录要点。

2. **PPT制作 (按分组)**
    - **封面**: 实验名称、组号、成员及分工。
    - **人体计算机组成**: 一页图文并茂地展示控制组和数据组的分工。
    - **指令集**: 列出本次实验定义的指令集。
    - **快速排序代码**: 贴上你们所依据的快速排序伪代码或真实代码。
    - **执行效果**:
        - 用几张关键照片展示排序过程（初始、某次分区后、最终）。
        - 嵌入快进的录像视频。
    - **思考题解答**: 针对指定的思考题，逐一进行回答和阐述。
        - **如何确保三个正确性?** (重点说明监督器、控制器分离，以及指令集的作用)。
        - **遇到的意外情况及应对?** (如：有同学理解错指令怎么办? 监督器如何纠错?)。
        - **计算机为何需要寄存器?** (提示：可以把正在比较的两个同学的位置索引，或基准值，看作是临时存储在“寄存器”中的数据)。
        - **为何需要随机选基准?** (讨论最坏情况，如初始队列基本有序时，选择第一个元素做基准的低效性)。
3. **班级汇报 (由班长准备)**
    - 汇总各组的亮点、共性问题和不同的思考。
    - 准备一个5分钟的总结性报告，向全班分享本次实验的收获与反思。

附录一：角色职责与行动指南

本文件为“人体快速排序计算机”实验中的每个角色提供了详细的职责描述和行动指南，请各位同学仔细阅读，明确自己的任务。

一、 控制组 (大脑中枢)

1. 监督器 (算法的守护者)

- **核心职责: 确保算法正确性**。你是团队中最理解快速排序算法的人。
- **实验前**:
    - [ ] **精通算法**: 反复阅读并完全理解快速排序的伪代码（见《实验核心要素》），特别是分区(Partition)的逻辑。能在纸上模拟一遍小型数据（如5-7个数字）的排序过程。
    - [ ] **准备工具**: 打印一份清晰的快速排序流程图或伪代码，作为你现场监督的“圣经”。
- **实验中**:
    - [ ] **全程紧盯**: 站在能看清全局的位置，观察“控制器”的每一步指令和“数据组”的每一次移动。
    - [ ] **同步标记**: 当“控制器”发出指令时，在你的流程图上同步标记当前执行到哪一步。
    - [ ] **果断纠错**: 一旦发现任何与算法不符的操作（如基准选择错误、比较逻辑错误、交换位置错误），立即大声喊“**停！**”。
    - [ ] **清晰解释**: 向团队解释错误发生在哪里，以及正确的操作应该是什么。
    - [ ] **指导回溯**: 指导团队回退到上一个正确的状态，然后继续执行。
- **实验后**:
    - [ ] **参与复盘**: 在讨论环节，分享在保证算法正确性方面遇到的挑战和经验。

2. 控制器 (唯一的指令发布者)

- **核心职责: 确保系统正确性**。你是“人体计算机”的CPU，所有操作指令都必须由你发出。

- 实验前:
    - [ ] **熟悉指令集**: 熟记《实验核心要素》中定义的每一条指令，并理解其确切含义。
    - [ ] **制定策略**: 与“监督器”沟通，确定基准（Pivot）的选择策略（例如，是选第一个、最后一个，还是随机选择？）。
- 实验中:
    - [ ] **发号施令**: 使用标准、清晰的指令（如“`COMPARE, A, B`”、“`SWAP, C, D`”）来指挥“数据组”。避免使用模糊的日常语言。
    - [ ] **保持串行**: 严格遵守一次只做一件事的原则。完成一条指令并确认“数据组”执行无误后，再发出下一条。
    - [ ] **掌控流程**: 明确当前正在处理哪个子队列，以及这个子队列的起始和结束位置。
    - [ ] **结果验证**: 排序全部结束后，从头到尾依次检查队列，向全队宣布“结果正确”或指出问题。
- 实验后:
    - [ ] **参与复盘**: 分享作为指令发布者遇到的困难，例如指令如何设计更高效，如何与团队更好地协作。

3. 监控器 (历史的记录者)

- **核心职责**: **记录实验过程，为汇报提供素材**。
- 实验前:
    - [ ] **准备设备**: 确保手机或相机电量充足，存储空间足够。
    - [ ] **明确拍摄重点**: 知道哪些是必须记录的关键瞬间。
- 实验中:
    - [ ] **拍摄关键帧**:
        - **初始状态**: 所有人按学号排好的原始队列（必须拍）。
        - **每次分区后**: 每当一个基准（Pivot）归位，队列被分成两部分时，立即拍照记录（非常重要）。
        - **最终状态**: 排序完成，所有人按身高排好的最终队列（必须拍）。
        - **意外/纠错瞬间**: 如果出现错误和纠正过程，尽量抓拍下来，这是宝贵的复盘材料。
    - [ ] **录制视频**: 在排序过程的某个阶段，录制一段1-2分钟的视频，展示团队协作和算法执行的动态过程。后期可以快进处理。

- 实验后:
    - [ ] **整理材料**: 将照片和视频分类整理，并分享给制作PPT的同学。
    - [ ] **挑选精华**: 挑选出最能体现算法思想和团队工作的照片用于PPT展示。

4. 计数器 (性能的分析师)

- **核心职责**: **量化算法的“计算量”**。
- 实验前:
    - [ ] **准备工具**: 准备好纸笔，制作一个简单的计数表格，或使用物理计数器。
    - [ ] **明确计数对象**: 只统计两个核心操作：**“比较”**和**“交换”**。
- 实验中:
    - [ ] **集中精神**: 仔细听“控制器”的指令。
    - [ ] **及时计数**:
        - 每当听到“`COMPARE`”指令，就在“比较”项下记一笔。
        - 每当听到“`SWAP`”指令，就在“交换”项下记一笔。
    - [ ] **保持准确**: 不要分心，确保不漏计、不重复计。
- 实验后:
    - [ ] **报告数据**: 向团队报告“比较”和“交换”的总次数。
    - [ ] **参与分析**: 这个数据是分析算法效率和讨论基准选择策略好坏的重要依据。

寄存器（可选角色）

- 目的：在现场把“寄存器”这一计算机概念可视化，作为短期存储基准值、索引或正在比较的两个元素，帮助理解寄存器在算法执行中的作用。
- 人数：建议 1–2 人（可从数据组临时调配，不必增加额外总人数）。
- 装备：每位“寄存器”携带索引卡与一个小白板或记号牌，用于记录当前的 pivot、i、j 等临时变量值。
- 实验中职责：
    - 在 `SELECT_PIVOT` 时，负责展示并“保存”基准值（在白板上写下 pivot 值并面朝全队），成为基准的临时载体。
    - 在分区过程中，按控制器或监督器指示，将当前 i/j 的索引和值展示出来，帮助全队清晰跟踪分区边界变化。

    - 在 `COMPARE` / `SWAP` 操作发生时，作为当前参与比较的“寄存器”物理代表，协助监控器记录操作顺序。
- 实验后：整理寄存器记录（白板照片或索引卡）并一并纳入监控材料，便于汇报与复盘。

二、 数据组 (运算器与存储器)

- **核心职责: 充当数据元素，被动、精确地执行指令。**
- **实验前:**
    - [ ] **理解规则:** 清楚地知道自己只能听从“控制器”的指令，不能自行移动或与他人商量。
    - [ ] **记住初始位置:** 记住自己按学号排列时的位置。
- **实验中:**
    - [ ] **保持专注:** 仔细听“控制器”是否在叫你的编号或位置。
    - [ ] **精确执行:**
        - 当被命令“比较”时，与指定的同学站在一起，等待“控制器”的判断结果。
        - 当被命令“交换”时，与指定的同学快速、准确地互换位置。
        - 当被命令“移动”时，立即走到指定的位置。
    - [ ] **保持静默:** 在没有指令时，保持在原地，不进行任何多余动作。你是“人体计算机”的一部分，你的行为代表了数据的存储和运算。
- **实验后:**
    - [ ] **分享体验:** 在讨论环节，分享作为“数据”被排序的感受，这有助于从另一个角度理解算法。

附录二：实验核心要素

本文件定义了“人体快速排序计算机”实验的核心技术规范，是实验成功的关键依据。

✓ 1. 人体计算机的组成

我们的“人体计算机”由 **26名同学** 构成，分为两个核心部分：

- **控制组 (Control Unit) - 4人**: 负责指挥和监控，是计算机的大脑。
    - **监督器 (Supervisor)**: 1人，保证算法执行的正确性。
    - **控制器 (Controller)**: 1人，发布所有操作指令。
    - **监控器 (Monitor)**: 1人，记录实验过程（拍照、录像）。
    - **计数器 (Counter)**: 1人，统计核心操作的执行次数。
- **数据组 (Data & ALU) - 22人**: 既是待排序的“数据”，也充当执行比较和交换的“算术逻辑单元”(ALU)。
    - 每位同学代表一个数据元素，其值为该同学的**身高**。
    - 初始顺序：按**学号**从小到大排列。
    - 最终目标：按**身高**从高到低排列。

✓ 2. 指令集 (Instruction Set)

为保证系统的串行性和正确性，“控制器”只能使用以下预设的指令与“数据组”进行通信。

指令 (Instruction)	操作数 (Operands)	描述 (Description)
`PARTITION`	`start_index`, `end_index`	**宣布**: 开始对从 `start_index` 到 `end_index` 的子队列执行一次分区操作。
`SELECT_PIVOT`	`index`	**选择**: 指定索引为 `index` 的同学作为本次分区的基准（Pivot）。

指令 (Instruction)	操作数 (Operands)	描述 (Description)
`COMPARE`	`index_A`, `index_B`	**比较**: 命令索引 `A` 和 `B` 的同学进行身高比较。控制器观察结果。
`SWAP`	`index_A`, `index_B`	**交换**: 命令索引 `A` 和 `B` 的同学立刻交换彼此的位置。
`PIVOT_RETURN`	`pivot_index`, `final_index`	**归位**: 命令基准（原在`pivot_index`）移动到其最终位置 `final_index`。
`HALT`	(无)	**暂停/结束**: 宣布所有排序操作完成或需要暂停。

✓ 3. 快速排序代码 (参考伪代码)

“监督器”和“控制器”应严格参照以下逻辑执行。这里提供一个经典的 Lomuto 分区方案作为参考。

```
// 主函数：对整个数组（队列）进行快速排序
function QuickSort(array, low, high):
  // 如果子队列至少有两个元素
  if low < high:
    // 1. 执行分区操作，找到基准的最终位置 pivot_index
    pivot_index = Partition(array, low, high)

    // 2. 递归地对基准左边的子数组进行排序
    QuickSort(array, low, pivot_index - 1)

    // 3. 递归地对基准右边的子数组进行排序
    QuickSort(array, high, pivot_index + 1)

// 分区函数：将数组分为“小于基准”和“大于基准”两部分
// 注意：我们的目标是身高从高到低，所以比较逻辑是反的
function Partition(array, low, high):
  // 1. 选择最后一个元素作为基准
  // 控制器指令: SELECT_PIVOT(high)
  pivot = array[high]

  // i 是“小于基准”区域的边界
  i = low - 1

  // 2. 遍历从 low 到 high-1 的所有元素
  for j from low to high - 1:
    // 控制器指令: COMPARE(j, high)
    // 如果当前元素 array[j] 比基准 pivot 更高 (即 "大于")
    if array[j] > pivot:
      // 将其与“小于基准”区域的下一个位置交换
      i = i + 1
      // 控制器指令: SWAP(i, j)
      swap(array[i], array[j])

  // 3. 将基准放到它的最终位置 (i+1)
  // 控制器指令: SWAP(i + 1, high)
  swap(array[i + 1], array[high])

  // 4. 返回基准的最终索引
  return i + 1
```

【重要提示】：

- 以上伪代码是按升序排列的。在我们的实验中，目标是**从高到低**排序。因此，在 `Partition` 函数中，`if array[j] > pivot` 的判断逻辑是正确的（将更高的放到左边）。
- 团队可以根据理解，采用其他分区方案（如 Hoare 分区方案），但必须在实验前统一，并告知“监督器”。

附录三：相关思考题解答

本文件旨在对实验提出的思考题进行深入分析与解答，帮助团队成员更好地理解实验的深层意义。

✓ 1. 你们如何确保三个正确性?

这是本次实验的核心考核点，我们通过**职责分离**、**流程标准化**和**物理记录**相结合的方式来确保。

- **确保结果正确性 (Result Correctness)**:
    i. **最终验证**: 排序完成后，“控制器”会从队首（身高最高）到队尾（身高最低）逐一进行相邻比较。
    ii. **物理见证**: 全体队员作为见证者，直观地确认最终队列是否符合身高递减的顺序。这是一个简单但有效的最终检查。
- **确保算法正确性 (Algorithm Correctness)**:
    i. **角色分离**: 这是关键设计。我们将“指挥”和“监督”的权力分开。“控制器”负责发号施令，“监督器”则手持算法流程图，像代码审查（Code Review）一样，独立地审查“控制器”的每一个指令是否符合快速排序的逻辑。
    ii. **“中断”机制**: “监督器”被授予最高权限，一旦发现指令与算法不符，可以立即喊“停”，形成一个“硬件中断”，强制程序暂停，进行纠错和回溯。
    iii. **标准化分区**: 团队在实验前就统一了分区方案（例如，统一采用Lomuto分区方案），并明确了基准选择策略，避免了因算法理解不一导致的混乱。
- **确保系统正确性 (System Correctness - Serial Execution)**:
    i. **唯一的指令源**: 我们建立了严格的命令链。“控制器”是唯一的指令发布者，“数据组”成员被严格要求只能被动地接收并执行来自“控制器”的指令。
    ii. **原子化指令集**: 我们设计的“指令集”（`COMPARE`, `SWAP`等）是原子化的，即每条指令都是一个不可分割的最小操作单元。“控制器”必须在一个指令被完全执行后，才能发布下一个指令，从而在宏观上保证了整个系统的串行执行。

iii. **禁止并行操作**: 明确禁止“数据组”成员之间自行交流、判断或移动。他们的行为完全由“控制器”的时钟信号（即指令）驱动，模拟了CPU的串行工作模式。

✓ 2. 实验过程出现了哪些意想不到的情况？你们是如何应对的？

(这是一个开放性问题，以下为可能情况的预演及应对策略)

- **意外情况1：指令理解错误**
  - **描述**: “控制器”发出 `SWAP(5, 9)` 指令，但第5号同学和第8号同学交换了位置。
  - **应对**: “监督器”或“控制器”在检查执行结果时会发现这个错误。此时，“控制器”会立即发出 `HALT` 指令暂停流程，然后重新发布一次 `SWAP(5, 9)` 指令，并口头强调是“五号”和“九号”，确保指令被正确执行。这个过程会被“监控器”记录下来，作为“鲁棒性”的体现。
- **意外情况2：基准选择导致效率低下**
  - **描述**: 在对一个已经接近有序的子队列进行排序时，我们仍然选择了第一个元素作为基准，导致分区严重不均，一边是0个元素，另一边是n-1个元素，使得算法退化成类似冒泡排序的 $O(n^2)$ 复杂度。
  - **应对**: “计数器”会发现某次分区的“比较”次数异常地多。实验暂停时，团队可以进行简短讨论。“控制器”和“监督器”可以决定在下一次大的分区操作中，尝试**随机选择基准**（例如，通过随机报数的方式）来优化性能，并观察“计数器”的读数变化。
- **意外情况3：场地干扰**
  - **描述**: 实验过程中，有其他非实验人员穿过我们的“数组”，导致队列混乱。
  - **应对**: “控制器”立即喊“停”。团队成员保持原地不动，“控制器”和“监督器”根据记忆和“监控器”的照片，快速恢复到干扰前的状态，然后继续实验。这体现了系统的“状态恢复”能力。

✓ 3. 计算机为什么应该有寄存器？快速排序实验的答案是什么？

- **通用解释**:

计算机的CPU（中央处理器）处理数据的速度远快于从主内存（RAM）中读取数据的速度。如果每次计算都直接读写内存，CPU将花费大量时间在等待数据上，造成巨大的性能浪费。

寄存器 (Register) 就像是CPU内部的“高速缓存”或“草稿纸”。它们是少量但速度极快的存储空间，直接集成在CPU核心旁边。CPU将当前最常用、最关键的数据（如循环变量、计算的中间结果、内存地址等）放在寄存器中，这样就可以瞬时访问，极大地提高了运算效率。

- **本实验中的体现**:
  在我们的“人体计算机”中，虽然没有物理的寄存器，但其**概念**体现在以下几个方面：
  i. **“控制器”和“监督器”的大脑**: 当“控制器”说“`COMPARE(i, j)`”时，`i` 和 `j` 这两个索引值就临时存储在了“控制器”和“监督器”的**短期记忆**里。他们不需要每次都去查阅完整的名单（相当于主内存），而是直接在大脑中处理这两个“变量”。这个“短期记忆”就扮演了寄存器的角色。
  ii. **正在操作的“人”**: 当两个同学被叫出来，站在中间进行“比较”时，这两位同学本身就成为了一个物理的、临时的“数据寄存器”。他们的数据（身高）被加载到“CPU”（控制器的眼睛和大脑）中进行处理。
  iii. **基准（Pivot）**：在一次完整的分区操作中，那个被选为基准的同学（或其身高值），在整个分区过程中都需要被反复访问。这个基准值就相当于存储在一个关键寄存器中，供所有其他数据与之比较。

  结论: 如果没有寄存器（即我们的大脑无法记住当前操作的是哪几个人），“控制器”每执行一步操作，都可能需要从头到尾重新确认每个人的位置和状态，这将使执行效率变得极其低下。寄存器解决了CPU与内存之间的速度矛盾，是现代计算机高性能的基石。

✓ 4. 快速排序算法为什么要有随机选择?

快速排序的平均时间复杂度是优秀的 O(n log n)，但它的**最坏情况时间复杂度是 $O(n^2)$**。这个最坏情况通常发生在**分区极其不平衡**的时候。

- **最坏情况如何发生?**
  想象一下，如果我们要排序一个已经完全有序（或基本有序）的数组，而我们**每次都选择第一个或最后一个元素作为基准**。
  - 例如，对 `[1, 2, 3, 4, 5]` 升序排序，选择 `1` 作为基准。分区

后，`1` 的左边是0个元素，右边是4个元素 `[2, 3, 4, 5]`。
    - 接下来对 `[2, 3, 4, 5]` 排序，选择 `2` 作为基准，同样，左边0个，右边3个。
    - ……以此类推。
    - 这样一来，每次递归调用只把问题的规模减小了1，总共需要进行 n 次递归，每次递归的遍历又是 O(n) 的，总复杂度就变成了 $O(n^2)$，性能与低效的冒泡排序相当。
- **随机选择如何解决问题?**

  通过**随机选择一个元素作为基准**，我们极大地降低了连续多次都选中最差基准（最大或最小值）的概率。
    - 在任何一次分区中，随机选择的基准有很大概率会落在数组的中间部分，从而使得分区相对平衡（例如，一边是 1/4，另一边是 3/4，甚至是理想的 1/2 对 1/2）。
    - 只要分区是相对平衡的，递归树的深度就能维持在 log n 的量级，从而保证了 O(n log n) 的平均时间复杂度。

  结论: 随机化是一种简单而极其有效的策略，它使得快速排序在面对各种输入数据时，都能以极高的概率表现出优秀的性能，避免了在特定数据模式下性能的灾难性下降。它用一个很小的随机成本，换取了算法整体的稳定性和高效性。