Solutions

Each answer shows the steps, then the **result**.

2.1

- any two of: CPU (time), memory/RAM, disk/storage, I/O devices

2.2

- gives each process a **fixed time slice** of CPU in turn, then moves it to the **back of the queue**

2.3

- **New, Ready, Running, Blocked, Terminated**

2.4

- saving the state of one process (to its PCB) and **restoring** another's, so the CPU can switch between them

3.1

- New → Ready (admit)
- Ready ⇄ Running (dispatch / time-out)
- Running → Blocked (I/O request) and Blocked → Ready (I/O done)
- Running → Terminated (exit)

3.2

- the process moves from **Running** to **Blocked**
- the scheduler then **dispatches another ready process** to the CPU while it waits

3.3

- each process has a **virtual address space** split into **pages**
- only the pages in use are kept in RAM **frames**, the rest stay in the **swap file** on disk
- on a **page fault** the OS swaps a page in (evicting one if needed), so the program can be larger than physical RAM

3.4

- too little RAM for the active pages, so the OS spends most of its time **swapping pages in and out** instead of doing useful work

3.5

- SJF runs the ready process with the **shortest expected run time** next
- benefit: the **lowest average waiting time** — short jobs are not stuck behind long ones