Solutions

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

2.1

- any two of: manages the hardware for programs; lets several programs share the hardware safely; provides services (files, network); provides a user interface

2.2

- **process** (or CPU) management

2.3

- gives each process the memory it needs and **keeps processes apart**
- uses virtual memory so the working set can exceed physical RAM

2.4

- it provides ready-made, tested routines, so the programmer **saves time** and avoids re-writing/re-testing code

3.1

- the hardware is complex and shared; if every program drove it directly they could clash or corrupt each other
- the OS provides a **safe, uniform interface** and shares the hardware fairly
- so programs are simpler and protected from one another

3.2

- any two, each described, e.g. **process management** — schedules CPU time and switches between processes
- **memory management** — allocates memory and keeps processes apart

3.3

(a) it moves the scattered pieces of fragmented files so each file's blocks are **together**

- that reduces seek time

(b) an SSD has no moving parts / no seek time, so reordering blocks gives no benefit (and wears the drive)

3.4

- the OS gives each program its **own block of memory** and checks every access
- an access outside a program's block is blocked, so one program cannot read or change another's memory

3.5

(a) any two: install and use **device drivers** so peripherals work; allocate devices to processes and queue their requests; manage the buffers between fast processor and slow

devices; handle interrupts from hardware

(b) any two: authenticate users with usernames and passwords; set and enforce **access rights** on files; keep an audit log of who did what; run or schedule backups; apply security updates

(c) the file system keeps a **directory structure**, and each student's work is stored in a different directory

- a file is identified by its **full path**, not just its name, so two identical names in different directories are different files
- the OS also holds each user's access rights, so one student cannot overwrite the other's file even by accident

(d) benefit: the routines are already written and **tested** by others, so development is faster and the code is likely to have fewer faults — encryption in particular is very hard to write correctly

- drawback: the programmer does not control the code and may not understand it fully, so a fault or a security weakness in the library becomes a fault in their program, and the library may be larger than needed or stop being maintained

(e) the kernel holds only what must run with full privilege and be available at all times

- a defragmenter is an occasional maintenance task that can run as an ordinary program, and keeping it out of the kernel keeps the kernel smaller and less likely to fail