1. A process waiting for disk I/O is –

   ☐ running

   ☐ ready

   ☐ terminated

   ☐ blocked

2. For the course set (7, 4, 1, 4 at time 0), the policy with the LOWEST average turnaround is –

   ☐ round robin q=2

   ☐ a tie

   ☐ FCFS

   ☐ SJF

3. Two threads each add 1 to counter = 5 with read-then-write. The unlocked interleaving R1 R2 W1 W2 leaves –

   ☐ 6

   ☐ 5

   ☐ 7

   ☐ 10

4. 16-bit virtual addresses with 256-byte pages give – virtual pages

   ☐ 16

   ☐ 64

   ☐ 256

   ☐ 1024

5. With 1 KiB blocks, 4 direct pointers and one 256-pointer indirect block, the max file is –

   ☐ 1 MiB

   ☐ 260 KiB

   ☐ 4 KiB

   ☐ 256 KiB

6. User code reaches kernel services only through –

   ☐ the TLB

   ☐ system calls

   ☐ the page table

7. Switching threads is cheaper than switching processes because threads –

   ☐ skip the registers

   ☐ run slower

   ☐ skip the address space

   ☐ need no scheduler

8. Under FCFS the set finishes at –

   ☐ 1, 5, 9, 16

   ☐ 16, 12, 11, 7

   ☐ 7, 11, 12, 16

   ☐ 5, 11, 13, 16

9. A spinning lock burns CPU; a mutex queue avoids it by –

   ☐ sleeping the waiter

   ☐ disabling the clock

   ☐ ignoring the lock

   ☐ sharing the lock

10. With VPN 2 mapped to frame 5, address 0x0204 translates to –

    ☐ 0x0504

    ☐ 0x5204

    ☐ 0x0025

    ☐ 0x0205

1. **blocked**

   Blocked waits on a device; ready waits only for the CPU.

2. **SJF**

   SJF averages 7.75 against FCFS 11.5 and RR 11.25.

3. **6**

   Both reads see 5; both writes store 6 – one increment lost.

4. **256**

   8-bit VPN after the 8-bit offset: 2 to the 8th is 256 pages.

5. **260 KiB**

   (4 + 256) x 1 KiB = 260 KiB.

6. **system calls**

   The mode bit walls the rings; syscalls are the only doors.

7. **skip the address space**

   Threads share the address space, so the switch skips it – only stacks and registers move.

8. **7, 11, 12, 16**

   First-come first-served: cumulative bursts 7, 11, 12, 16.

9. **sleeping the waiter**

   The blocked waiter sleeps in the OS queue instead of spinning.

10. **0x0504**

    frame x 256 + offset: 5 x 256 + 4 = 0x0504.