Problem 1: OS Overview and Boot Process

1.1 (3 pts) What are the THREE roles of an operating system? For each role, give an example as well.

1.2 (2 pts) In designing an OS, there are a number of criteria to consider such as reliability, portability and performance. If you were to design an OS for a game console, out of these three criteria, what is **the least important** to consider? Why?

1.3 (2 pts) How does BIOS detect if a given disk is bootable?

1.4 (2 pts) Explain briefly how to print out a character in a bootloader

Problem 2: Kernel and system calls

2.1. (4 pts) What is dual-mode of operation? Why is it important?

2.2 (3 pts) Name THREE different events that trigger a transition from user mode to kernel mode.

2.3 (3 pts) Name THREE system calls that you use in your shell project.

Problem 3: Processes and Signals

3.1 (2 pts) How many processes in total are created by the following program?

```
#include <stdio.h>
#include <unistd.h>

int main()
{
   /* fork a child process */
   fork();

   /* fork another child process */
   fork();

   /* and fork another */
   fork();

   return 0;
}
```

3.2 (2 pts) Normally, in a shell implementation, you call **fork()** then **exec()** to start a new process. What would happen if we call **exec()** then **fork()** like the following:

```
shell (..) {
  ...
  exec (cmd, args);
  fork();
  ...
}
```

What is the impact of this change to the shell, if any? (Explain)

3.3 (3 pts) Name <u>THREE</u> different ways a process may react to a signal delivered by the kernel.

3.4 (1 pt) What is the UNIX command for sending a signal?

3.5 (4 pts) What is the exit status of a process? How does the parent process retrieve this information?

Problem 4: Threads, Synchronization and Deadlocks

4.1 (6 pts) Consider a threaded program that is having an unknown performance issue. Give a possible cause and pick a resolution for each of the scenarios below.

a) CPU Usage: 50% | Disk usage 50%

Possible cause: ______________________________
Resolution (pick one): Get faster CPU / Get faster disk / Use more threads

b) CPU Usage: 100% | Disk usage 20%

Possible cause: ______________________________
Resolution (pick one): Get faster CPU / Get faster disk / Use more threads

c) CPU Usage: 20% | Disk usage 100%

Possible cause: ______________________________
Resolution (pick one): Get faster CPU / Get faster disk / Use more threads

4.2 (6 pts) In a social media application, it is common to have a “follow” functionality where one user is following and being followed by other users. Consider the following code:

```
typedef struct {
  int user_id;      // unique user id
  int n_follows;    // # of users this user is following
  int n_followed; // # of users that is following this user
  mutex lock;      // mutex for locking this structure
  …
} user_t;

// user a requests to follow user b
void perform_follow(user_t *a, user_t *b) {
  ...
  a->lock();
  b->lock();
  a->n_follows++;
  b->n_followed++;
  b->unlock();
  a->unlock();
  ...
}
```

a) When `perform_follow()` is called by many threads simultaneously, is it possible for the application to have a deadlock? How?
b) If so, can you suggest a modification to the code to eliminate any deadlock possibility.

4.3 (2 pts) For each of the following situations, indicate whether **kernel threads** or **user-level threads** are more suitable.

a) A multi-threaded video player application where each thread reads different parts of the video from disk.
b) A multi-threaded game where each thread is rendering different part of the screen by using data cached in memory (no I/O operation needed)

4.4 (5 pts) Suppose two threads execute the following C code concurrently, accessing shared variables `a,b,c`:

Initialization	*Thread 1*	*Thread 2*

`int a = 4;`
`int b = 0;`
`int c = 0;`
	`If (a < b) {`
	`  c = b - a;`
	`} else {`
	`  c = b + a`
	`}`
		`b = 10;`
		`a = -3;`

What are the possible values for c after both threads complete? You can assume that reads and writes of the variables are atomic, and that the order of statements within each thread is preserved in the code generated by the C compiler.

Problem 5: CPU Scheduling

5.1 (4 pts) A common scheduling algorithm is First-Come-First-Serve (FCFS) where jobs are assigned to CPU based on their arrival times. Describe a potential issue with FCFS algorithm.

5.2 (4 pts) In Shortest-Remaining-Time-First (SRTF) scheduling, the algorithm picks a job with the smallest amount of time remaining to schedule first. In practice, it is difficult to know the remaining time of a process in advance. What can the OS do to approximate the remaining time of a given process?

5.3 (3 pts) In round-robin scheduling, it is common to use a fixed-sized quantum.

a) What is a quantum?
b) What is a benefit of using a small quantum?
c) What is a benefit of using a large quantum?

Problem 6: Protection and Security

6.1 (4 pts) Explain what is the **principle of least privilege.** Give a concrete example exercising the principle of least privilege. You may use non-computer related examples.

6.2 (4 pts) For each of the following, indicate whether an **access list** or a **capability list** is being used and argue the weakness of the method.

a) During the covid-19 situation, many buildings set up temperature screening stations at their entrances. Those who already pass the screening will receive a sticker and can enter the building.

b) Every time someone wants to enter the Science Division, he or she must tap your ID card on the scanner.