United International University
Department of Computer Science and Engineering

CSI 309/CSE 4509: Operating System Concepts/Operating Systems

Midterm Examination: Summer 2023

Total Marks: 30 Time: 1 hour 45 minutes

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

Answer all the questions. Numbers to the right of the questions denote their marks.

1. Consider the set of 5 processes whose arrival time and burst time are given in the table 1. CPU follows the Shortest Remaining Time First algorithm.
   [**Assume that, multiple I/O request can be overlapped or served simultaneously**]

Table 1: Process with Arrival Time and Burst Time

Process	Arrival Time	Burst Time(ms)		
		I/O	CPU	I/O
P0	0	2	5	1
P1	1	0	4	3
P2	3	4	1	2
P3	2	1	3	0
P4	7	3	2	2

   (a) Draw the Gantt Chart. [3]

   (b) Compute the average response time with CPU utilization. [1+1]

   (c) "With the largest Time Quantum 5 ms, Round Robin gives the same response time as FCFS gives."-Justify the statement for the given scenario. [5]

2. (a) Find the output of the following code: [5]

```
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
void main(){
    pid_t ret1 = fork();
    pid_t ret2 = fork();

    if(ret2 == 0) {
        printf("Os is the worst!\n");
    }
    else if (ret1 == 0 && ret2 > 0) {
        wait(NULL);
        printf("OS is okay!\n");
    }
    else if (ret1 > 0 && ret2 == 0){
        wait(NULL);
        printf("OS is fun!\n");
    }
    else if (ret1 > 0 && ret2 > 0){
        wait(NULL);
        printf("OS is confusing!\n");
    }
    else {
        wait(NULL);
        printf("OS is the best!\n");
    }
}
```

[**Hint: wait(NULL) suspends the execution of the parent process until the child process terminates**]

(b) Imagine there are 5 processes running concurrently: A, B, C, D, and E. The Burst times of these processes are: A(3 Seconds), B(4 Seconds), C(2 Seconds), D(3 Seconds), and E(2 Seconds). While A was running, a chain of interruption was raised by B, C, D, and E at various times t, i.e. process B raises interruption at **t=1**, C raises at **t=3**, D at **t=5**, and E at **t=9**.

If processes B and D are Non-Maskable and C and E are Maskable, then draw the Control flow Diagram of the CPU for each time step.

[**Ignore the time taken by the Interruption Handler and Context Switching. The first one is done for you in figure 1.**] [3]

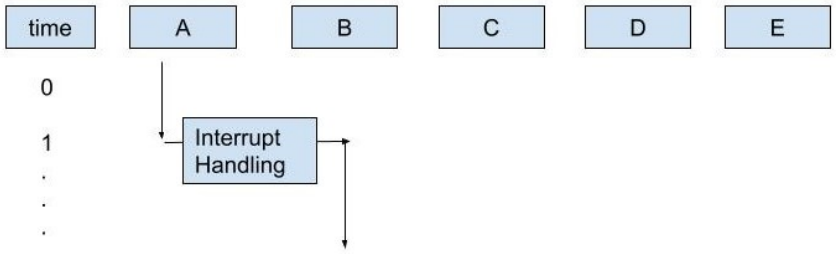

Figure 1: Part of Control Flow Diagram for Ques 2b

(c) Can we create a program that can run without an operating system? [1]

(d) What mechanism will we need if we want to run multiple operating systems simultaneously on a single hardware? [1]

3. (a) For the following code, find out where the race condition arises. Then modify the code to resolve those potential race conditions. [3 + 2]

```
#include <iostream>
#include <thread>
#include <random>
#include <chrono>
#include <mutex>
#include <semaphore.h>

using namespace std;

int collectibles = 200, artifacts = 0, artifacts_generated = 0;
int playerScore = 0, aiScore = 0;

void playerCollect() {
    while (collectibles > 0) {
        collectibles--;
        playerScore += 5;

        if (artifacts > 0) {
            artifacts--;
            playerScore += 50;
        }
    }
}

void aiCollect() {
    while (collectibles > 0) {
        collectibles--;
        aiScore += 5;

        if (artifacts > 0) {
            artifacts--;
```

```
            aiScore += 50;
        }
    }
}

void spawnArtifact() {
    // Code for random number generation
    random_device rd;
    mt19937 gen(rd());
    uniform_real_distribution<double> dis(0.0, 1.0);

    while (collectibles > 0) {
        if (dis(gen) < 0.2) {  // 20% chance of spawning an artifact
            artifacts++;
            artifacts_generated++;
        }
        this_thread::sleep_for(chrono::seconds(2));
    }
}

int main() {
    srand(static_cast<unsigned int>(time(nullptr)));

    thread playerThread(playerCollect);
    thread aiThread(aiCollect);
    thread spawnerThread(spawnArtifact);

    playerThread.join();
    aiThread.join();
    spawnerThread.join();

    return 0;
}
```

Use table-2 as a reference for the usage of mutex and semaphore.

Table 2: A table showing methods for declaring, locking, and unlocking mutex and semaphore.

Variable	How to Declare	How to Lock	How to Unlock
Mutex	mutex sharedLock	sharedLock.lock()	sharedLock.unlock()
Semaphore	sem_t sharedSem	sem_wait(&sharedSem)	sem_post(&sharedSem)

[You do not need to write the full code. Write only the modifications.]

(b) Imagine, you have been transported back in time and got a job as a teaching assistant in a 90s classroom. They did not have individual computers for every student in a lab back then. Students had to bring their codes in punched cards and had to execute their codes on a shared computer. Your responsibility is to maintain this queue for the computer. Which one, between a mutex and a binary semaphore, is more applicable here? Explain in detail. [3]

(c) Draw the state diagram of a process. [2]