# National University of Computer and Emerging Sciences

**FAST School of Computing** **Spring 2020** **Islamabad Campus**

# CS-220: Operating System

## Remote Final Exam

**Attempt Time: 3 Hours**
**Submission (on LMS and through email) Time: 15 minutes**

Tuesday, 14th July, 2020

## Course Instructor(s)

Faisal Cheema, Sidra Khalid, Muhammad Adnan Tariq, Hasan Mujtaba

## Total Marks: 110

### Instructions:

1. The final exam will be attempted offline in the student's own handwriting (in readable way, if we can't understand it, we won't mark it).
2. The students will use A4 size blank white sheets to attempt the exam (portrait format unless a diagram or table requires landscape). Each sheet of the A4 size paper **MUST** have the Roll Number, Name, the course code, name of the course and Signature of the student at the top of **EACH** sheet.
3. Students will use cam-scanner, MS lens, or an equivalent application to scan and convert their hand-written answer sheets into a **SINGLE** pdf file (keeping the correct order of pages and question numbers), which they will submit on LMS and **MUST also** email to the email address (of the concerned course/lab instructor) which will be provided. They will be given 15 minutes (after the 3 hours attempt time) for this purpose. All students must use the standard file name format (Full course code - Roll number e.g. CS-305-17i-0123). Submissions after 30 minutes may not be accepted. **Try to submit soon after 3 hours of attempt time and do not wait for 15 minutes to be elapsed.**
4. For proven cheating/ plagiarism, student will get an F grade even if the student had opted for S/U grade, and the case will be referred to DDC (Department's Disciplinary Committee). Instructors will conduct vivas of randomly selected students or in case of doubt (significantly different attempt as compared to past performance in the course or matching attempt with other students). Plagiarism includes sharing an attempt to other students (copy providing). Students who are not able to satisfactorily answer instructor's questions (based on the exam as well as slightly lateral but related concepts) during viva will also be considered as plagiarism cases.
5. Students should carry a clean scanning that is free from any marks/stains etc.
6. Paper is divided into four independent sections, clearly mark the section and question number on top of the sheet to avoid confusion. Do not attempt questions from different section on the same sheet.

| | S-1 | S-2 | S-3 | S-4 | Total |
|---|---|---|---|---|---|
| Marks | 50 | 20 | 20 | 20 | |

# Section 1

**Question 1 [20 Marks]**

Consider the following program

```
const int n = 52;
int Covid;

void sum(){
    int count;
    for (count = 1; count ≤ n; count++){
        Covid++;
    }
}

void main(){
    Covid = 0;
    Parbegin(sum(), sum())     // two Processes start concurrently
    Print(Covid);              // after termination of total
}
```

1. What will be the minimum and maximum value of shared variable "Covid". Note that processes can execute at any relative speed. Moreover, to increment a value it has to be loaded into a register by a separate machine instruction. Likewise, the incremented value has to be stored in main memory by a separate machine instruction. Justify your answer by providing the complete execution scenario. [10 marks]

2. If an arbitrary number of these processes (i.e., N processes) are allowed to run in parallel, what effect will this modification have on the minimum and maximum value in the above question? Justify your answer by providing the complete execution scenario. [10 marks]

**Question 2 [30 Marks]**

Consider the following program where two processes are executing the function sum() concurrently and prints final value of the shared variable b.

```
Init:
    int a= 0;                  // a is a shared variable
    int b= 0;                  // b is a shared variable
    Binary Semaphore S = 1;    // S is binary semaphore

void sum(){
    for (int i=0; i ≤ 5; i++){  // i is a local variable
        a = a + 1;
        wait(S);
        b = b + a;
        signal(S);
      }
}

void main(){
    Parbegin(sum(), sum()) // two Processes start concurrently
    Print(b);              // after termination of sum
}
```

Assumptions: In each assignment, the values of the variable is first loaded into CPU register by a separate machine instruction, afterwards CPU performs the respective arithmetic operation and then writes the result to the respective variable (in main memory). Moreover, processes can execute at any speed and can be preempted by OS at any time in between consecutive machine instructions.

1. Determine the maximum possible value for b and state an according execution of the program. [10 marks]

2. Determine the minimum possible value for b and state an according execution of the program that produces this minimum value. [20 marks]

# Section 2

**Question [20 Marks]**

Suppose in a sports club, there is a sports instructor who teaches the art of juggling with juggling pins. This course is a basic training course where a student is taught to juggle two pins in a one-one session. After the training the student can practice juggling for a particular time period. Suppose there are three students who plan to practice juggling at the same time. But there are only 4 juggling pins available in the room. To start practice, each student requires exactly 2 pins, as shown below:

```
while (!practice_complete())
{
acquire_one_juggling_pin(); // Acquires 1st juggling pin
acquire_one_juggling_pin(); // Acquires 2nd juggling pin
juggle();
release_one_juggling_pin(); // Releases 1st juggling pin
release_one_juggling_pin(); // Releases 2nd juggling pin
}
```

1. Can this practice plan lead to deadlock? If yes, explain with an example scenario. If no, explain your answer. [10 marks]

Suppose the students have passed the beginner course in juggling. They now join the advance group for further training and practice. In the training sessions now, there are arbitrary number of jugglers, N, learning to juggle. Furthermore, each juggler may require different number of juggling pins depending on acquired skill level. Another, parameter to consider is P i.e. the number of juggling pins that would satisfy the need of all jugglers in the practice room. For example, in a practice session, three jugglers require 2 juggling pins, one requiring 3 pins whereas another juggler needs 4 pins to juggle. That results in P = 13. The changed practice protocol is shown as:

```
int num_pins = how_many_pins_needed();
while (!practice_complete()) {
	for (int s = 0; s < num_pins; ++s) {
		acquire_one_juggling_pin(); // Acquires a juggling pin
	}
	juggle();
	for (int s = 0; s < num_pins; ++s) {
		release_one_juggling_pin(); // Releases a juggling pin
	}
}
```

2. What is the smallest number of juggling pins (in terms of P and N) needed to ensure that deadlock cannot occur? Explain your answer with an example scenario in terms of some particular P and N values of your choice (do not use the above given example in the description). [10 marks]

# Section 3

**Question [20 Marks]**

Since this description of the system is fairly vague, you will have to make certain assumptions to answer the questions. Read the questions below and before you start you answer clearly list/state all assumptions that you will be making to answer the question. This part is crucial, no two students attempting the question on their own will come up with the same assumptions. Do not help any other student with their answer as this would likely lead to plagiarism which would lead to an F grade in the OS course.

Consider that you are designing the OS for a mysterious client. The full functionality of the OS is never elaborated to you; however, you are provided the following bits and pieces of information are available to you.

1. Number of processes in execution is fixed, i.e. there are always X number of processes in execution.
2. These X processes request the same number of resources Y every time.
3. When and how a process would request a resource is never known.
4. User is allowed to execute new task (via a custom shell) in which case an older task has to be stopped.

**Given your design decisions, the requirement of the system; list and elaborate on all possible mechanism which can be used to detect race conditions during runtime in this system. Marks will be awarded based on the efficiency and relevancy of the solution presented.**

A solution that can detect more scenarios than others will be considered better, keeping view that the underlying assumptions and design are reasonable and implementable.

Simply stating race conditions cannot be detected at runtime will be awarded with zero marks. Marks will be deducted for writing unnecessary and irrelevant details like the definition of race conditions etc.
Bonus: (only in this section, do not attempt this with answers of other sections)
Share an achi thought, not a quote or something you read off the internet, share your own thought in 2-3 lines. (max 2 marks)

*Hint: Given the open-ended and tricky nature of this question, it would be advisable to balance your time, rather than going for a perfect answer that handles all possible scenarios (which is probably impossible), save time and go for a practically do able answer that will work in most scenarios.*

# Section 4

**Question [20 Marks]**

You are given with a variation of an OS called Linux_beta where three processes A, B and C are used to print a file. Processes A, B and C are additionally using two shared buffers to store the text of file so buffers are being used for coordination and synchronization.
Below steps will be performed for printing a file in Linux_beta OS:

1. Process A reads the file data from the disk to Buffer 1
2. Process B copies the data from Buffer 1 to Buffer 2
3. Process C takes the data from Buffer 2 and print it.

Below is the skeleton pseudo code of the three processes, how these three processes maintain synchronization to access buffer and printing file using semaphores. Write missing pseudo code.

```
Process_A () {
 while(1) {



read(next_file(), Buffer_1);



 }
}
```

```
Process_B () {
 while(1) {



 copy(Buffer_2, Buffer_1);









    }
}
```

```
Process_C () {
  while(1) {



   print(Buffer_2);



  }

}
```

*Good luck and best wishes.*