National University of Computer & Emerging Sciences, Karachi
CS-Department
FAST School of Computing
Lab Final Examination Fall 2023

Course Code: CL2006	**Course Name: Operating Systems Lab**
Instructor Names: Mr. Muhammad Monis	
Student Roll No:	**Section No:**

General Instructions: Carefully read the following instructions before attempting the paper.

- Except your Roll No and Section, **DO NOT WRITE** anything on this paper.
- In case of any ambiguity, you may make assumptions, but your assumption must not contradict any statement in the question paper.
- **DON'T** share your program, if your code is matched to any member of your class, both will get **straight F** in the course without asking who shared or who magically copied.

Submission Instructions:

- You must comment your student ID on top of each file. (Line#1 of your code).
- Name the .c file for each question according to Roll_No e.g. **k22-xxxx_Q1.c, k22- xxxx _Q2.c** etc.
- Create a ZIP folder of all your solutions and copy it in the local storage with the title **K22-xxxx_A.**
- Submission are on local storage that can be accessed using win+r keys and entering \\172.16.5.43 address in the dialog box.
- Enter your username as **khifast\K22xxxx** and its assigned password (Default is **Fast1234**).
- Zip folder needs to be pasted in the "**Exam Folder\teacherName\Your_Roll_No**" folder

Total Time: **110 minutes** **Maximum Points**: **50**

Question No 01(Shell Script + Scheduling Algorithm) **[Marks: 10 Points]**

You are tasked to create multiple shell scripts with codes containing the following examples.
1. Calculate the Factorial, 2. Calculate the prime number, 3. Create/Delete Folders (3 points) .
2. Create a .C file Containing an scheduling algorithm to execute the tasks (2 points).
3. Scheduling has to be done in a way each task has an equal execution time but the time needs to be dynamic based on the initial task being scheduled.
4. Calculate this time based on the number of loops inside of the program (The greater the higher the time). (4 points)
5. Execute the scripts (order should be Fact-Prime-Create scripts) waiting time can be taken as an assumption (1 point).

Question No 03(OpenMP) **[Marks: 10 Points]**

Assume you have an array of 20 integers, and your task is to find the largest and smallest elements in the list, along with their respective index positions in the original array. To optimize the solution and improve performance, you decide to leverage the power of OpenMP libraries to divide the workload among multiple threads. After finding the largest and smallest elements, you need to output their values and index positions. In the event if the user presses the SIG signal the program should halt and display the current maximum and minimum values it was able to find (even if it is incorrect). After halted the program should ask the user to further continue or leave the program. If continued turn the signal that is changed to default and resume the program with the array it was using to find the largest and smallest values.

S

Question No 03(Fork System Call+Inter-Proceess Communication) [Marks: 15 Points]

Consider the tree above in the Figure. The nodes in these are processes (i.e. Process 3 will send a message to process 2 and process 5 and process 2 will send a message to process 1 and process 5 will send message to process 4)

1. Create 5 fork child process in this tree configuration (2 point).
2. Create another file which will receive the message corresponding to the process child (i.e. The receiver file should receive process 2 and 5 if process 3 is started first) (4 points).
3. When receiving the message sent a message received back to the sender (2 point).
4. Once all the nodes are traversed send one last message to the sender which contains a DFS traversal (any method) of the tree to the receiver and print the nodes (4 points).
5. Use the appropriate IPC method (3 points).

Question No 03(Semaphores) [Marks: 15 Points]

Write a C program that simulates a race between n runners, with each runner represented by a separate thread. The program should ask the user for the number of runners, the length of the racetrack, and the speed of each runner. Assume that the runners take turns running in bursts. For example, Runner#1 will run according to their speed. Then Runner#1 will stop and Runner#2 will continue, and so on. At any one time, only 1 runner will be running and the rest will be idle. The program should create a thread for each runner and use a semaphore to ensure that only 1 runner runs at any one time. Each runner's speed will vary dynamically during the race based on random events, simulating real-life scenarios. The threads should race towards the finish line, and the program should output the order in which the runners finish and their respective times. Additionally, the program should determine and output the overall winner of the race.

Instructions:

1. Implement a C program that takes user input for the number of runners, the length of the racetrack, and the initial speed of each runner.
2. Create a separate thread for each runner participating in the race.
3. Use a semaphore to ensure that only 1 runner runs at any one time.
4. Simulate dynamic changes in each runner's speed during the race. At random intervals, update a runner's speed with a randomly generated number, to simulate real-life scenarios.
5. Each runner should race towards the finish line by advancing a certain distance based on their current speed.
6. Output the order in which runners finish the race and their respective times.
7. Determine the overall winner of the race based on the fastest finishing time and print the winner's details.