Operating Systems, Evaluation 1
CS30002, Spring 2021

25th February, 2022, 9:00 – 10:10 am
Full marks: 40
Attempt ALL questions

IMPORTANT INSTRUCTIONS

Taking the exam: You need to log into Google Meet (exam room links have been given earlier), keep your video on during the test so that we can monitor you during the exam. YOU HAVE TO USE PEN AND PAPER TO GIVE THE EXAM.

Decorum: Throughout the examination, you are strictly expected to have your cameras on, directing towards your work-space including yourself. Arrange your laptops/desktops/mobiles beforehand to save time during the examination. You should be visible on camera even while scanning/uploading your answer-script. Disconnecting video for a long duration will be grounds for suspecting malpractice. You need to keep your workplace and your hands visible to us. However, avoid the visibility of your answers to the rest of students. **Once you open your question paper, refrain yourself from using your PC/laptop/mobile for searching for anything or typing during the exam.**

Submission: You can do either of two things -- (i) take pictures of your answer script pages, name the pictures page1.jpg, page2.jpg, page3.jpg, etc., zip the pictures and upload the zipped file via CSE Moodle, or (ii) Put all the pages sequentially in a pdf file and upload the pdf to CSE Moodle.

Submissions must be through the course Moodle, by 10:10 am (according to the Moodle clock). The actual exam will be from 9:00 to 10:00 am and you have a dedicated 10-minute slot to upload your answer-script to Moodle. If you miss this deadline for Moodle submission, you need to email your submission (with OS Test - <Roll No> as subject) to TA Kousshik Raj (kousshikraj.raj@gmail.com) within 10:20 am; there will be a penalty of 20 marks for such late submissions. No submission will be allowed after 10:20 am. If any submission reaches the TA's mailbox after 10:20 am (according to the timestamp of the email), it will be rejected.

Policies: Note that, if we face problems with your answer script, e.g., cannot open your submitted zipped file, cannot read the text in pictures (due to bad resolution), cannot determine the page order from the file names (or if the pages in the pdf are jumbled up), it will affect your marks.

Malpractice: If any group of students is found to have similar answers/working in their answer sheets, ALL of them will receive the maximum penalty with no grace. We will not distinguish between who supplied answers and who copied; everyone involved will receive the maximum penalty. We expect you to NOT take help from the internet, your copies, textbooks, slides or video recordings during the exam. Note that this is NOT an open-book exam. Also, you should NOT discuss answers with anyone during the scheduled exam time. If found otherwise, you will be strictly penalized.

Tip: Install Adobe Scan app on your mobile phone to make the whole process easier. In that case, your laptop/PC acts as a camera, while you are using your mobile for checking the questions,

scanning and uploading the answers. [This is only a suggestion that can make things simpler for you, not a necessity. You are free to use other devices for scanning, e.g., if you have a scanner available. Note: scanning should also be done in front of the camera.]

PLESE WRITE YOUR NAME AND ROLL NO. ON THE TOP OF THE FIRST PAGE OF YOUR ANSWER SCRIPT. WE WILL NOT EVALUATE YOUR ANSWER SCRIPT WITHOUT IT.

Question 1: **[3 + 2 + 4 = 9 marks]**

You are the chief developer of a company which is going to launch your new OS for their proprietary handheld devices. While considering scheduling algorithms you stumbled upon the idea of *lottery scheduling* which is a probabilistic scheduling algorithm (with a goal to improved fair share of processes). The idea is actually simple---Processes are each assigned some lottery tickets (each with a unique number). Then the scheduler draws a random ticket number to select the next process (whoever owns the specific numbered ticket will execute next). The distribution of tickets *need not be uniform*; So, granting a process more tickets provides it a relative higher chance of selection.

1.1. You have two ways to implement lottery scheduling (i) keeping one ticket specific list of processes in kernel or (ii) For each process keep a list of tickets in PCB. Which of these approaches is more efficient keeping in mind the working on lottery scheduling? Why (no marks without explanation)

1.2. Theoretically, is *starvation* possible in lottery scheduling? Why or why not? (no marks without explanation)

1.3. In lottery scheduling there are two ways to distribute the tickets to processes ----user assigns tickets to their processes (from a fixed pool given to a user) and kernel assigns tickets. Write one advantage and one disadvantage for each of these ways.

Question 2: **[5 × 2 = 10 marks]**

Answer the following in brief.

2.1. "*The primary objective of multiprogramming is to minimize user response time, while the primary objective of time sharing is to maximize processor utilization.*" Justify or contradict.

2.2. Is it possible to find a Unix process (which is already done executing) to have never been in a state of zombie or orphan? Why or why not?

2.3. State four events that may lead to process context switch in a time-sharing operating system.

2.4. Justify with reasons whether the following CPU scheduling algorithms can result in process starvation: (A) first-come first-serve, (B) shortest-job next.

2.5. Why is thread scheduling faster than process scheduling?

Question 3: **[3 + 3 + 4 = 10 marks]**

Answer the following.

3.1. Consider the following pseudo-code for a process **Pi**, where "**shared boolean flag[2]"** is variable declared in shared memory, initialized as: **flag[0] = flag[1] = FALSE;**

```
P (int i) {                    // i=0 for P0, i=1 for P1
  while (TRUE) {
    flag[i] = TRUE;
    while (flag[(i+1) % 2) == TRUE);
      < Critical Section >
    flag[i] = FALSE;
      < Remainder Section >
  }
}
```

Explain if this code solves the critical section problem for two processes **P0** and **P1**.

3.2. Suppose that an instruction swap(int R, int *mem) is added to a processor that swaps the contents of a register R and a memory location mem in a single indivisible step. Suggest a solution to the critical section problem using this instruction.

3.3. Write a code segment in C/C++ where a parent process creates a shared memory segment, allocates an integer array X of size 10, and populates it with random integers. It then creates two child processes C1 and C2. C1 will update the array by subtracting the average of the numbers from each of the numbers (i.e. X[i] = X[i] – average(X)). C2 will wait for 1 second, and then print the contents of the array X. Finally, the parent process will remove the shared memory segment.

Question 4. **[3 + 3 = 6 marks]**

Consider a scenario where a CPU has to run 5 processes with different arrival and burst times. Calculate the *average waiting time* for the given scenario with the following scheduling algorithms- (i) non-preemptive SJF (ii) Round Robin Scheduling (time quantum = 5ms)

Process	**P1**	**P2**	**P3**	**P4**	**P5**
Arrival Time	0	0	4ms	6ms	8ms
Actual Burst Time	8ms	2ms	40ms	8ms	6ms
Expected Burst time	10ms	5ms	20ms	5ms	10ms

Question 5: **[2.5 + 2.5 = 5 marks]**

Justify with reasons whether the following statements are true or false (no marks without justification)

5.1. The non-preemptive SJF algorithm minimizes the average waiting time of a set of running processes, assuming that all processes arrive at time t = 0.

5.2. The Swapped-Out states in a process state transition diagram are required to keep track of the processes that do not require CPU time.