PART-I (Check/Build your Concepts)

Problem 1
Suppose that a scheduling algorithm (at the level of short-term CPU scheduling) favors those processes that have used the least processor time in the recent past. Why will this algorithm favor I/O-bound programs and yet not permanently starve CPU-bound programs?

Problem 2
Why is it important for the scheduler to distinguish I/O-bound programs from CPU-bound programs?

Problem 3
Explain the working of Multi-level and Multi-level Feedback Queue and Differentiate between them? What is the benefit of using Multi-level Feedback Queue?

Problem 4
Consider a system implementing multilevel queue scheduling. What strategy can a computer user employ to maximize the amount of CPU time allocated to the user's process?

Problem 5
Which of the following scheduling algorithms could result in starvation? Explain your answer.

a. First-come, first-served
b. Shortest job first
c. Round robin
d. Priority

Problem 6
Explain the differences in how much the following scheduling algorithms discriminate in favor of short processes:

a. FCFS
b. RR
c. Multilevel feedback queues

Problem 7
Discuss how the following pairs of scheduling criteria conflict in certain settings.

a. CPU utilization and response time
b. Average turnaround time and maximum waiting time
c. I/O device utilization and CPU utilization

Problem 8
Consider a set of n tasks with known runtimes R1, R2…R(n) to be run on a uniprocessor machine. Which of the processor scheduling algorithms will result in the maximum throughput?

Problem 9
What advantage is there in having different time-quantum sizes on different levels of a multilevel queuing system?

Problem 10
Consider a system running ten I/O-bound tasks and one CPU-bound task. Assume that the I/O-bound tasks issue an I/O operation once for every millisecond of CPU computing and that each I/O operation takes 10 milliseconds to complete. Also assume that the context-switching overhead is 0.1 millisecond and that all processes are long-running tasks. Describe the CPU utilization for a round-robin scheduler when:

a. The time quantum is 1 millisecond
b. The time quantum is 10 milliseconds

Problem 11

Many scheduling algorithms are parameterized. For instance, the round-robin algorithm requires a parameter to indicate the **time quantum**. The multi-level feedback (MLF) scheduling algorithm requires parameters to define the **number of queues**, the **scheduling algorithm for each queue**, and the **criteria to move processes between queues** (and perhaps others. . .). Hence, each of these algorithms represents a set of algorithms (e.g., the set of round-robin algorithms with different quantum sizes). Further, one set of algorithms may *simulate* another (e.g., round-robin with infinite quantum duration is the same as first-come, first-served (FCFS)). For each of the following pairs of algorithms, answer the following questions:

1. **Priority scheduling and shortest job first (SJF)**
   a) State the parameters and behavior of priority scheduling
   b) State the parameters and behavior of SJF
   c) Can SJF simulate priority scheduling for all possible parameters of priority scheduling? (How or why not: State how to set SJF scheduling parameters as a function of priority scheduling parameters or explain why this cannot be done.)
   d) Can priority scheduling simulate SJF for all possible parameters of SJF?(How or why not?)
2. **Multilevel feedback queues and first come first served (FCFS)**
   a) State the parameters and behavior of multi-level feedback queues
   b) State the parameters and behavior of FCFS
   c) Can FCFS simulate multi-level feedback for all possible parameters of multi-level feedback?
   d) Can multi-level feedback scheduling simulate FCFS for all possible parameters of FCFS?(How or why not?)
3. **Priority scheduling and first come first served (FCFS)**
   a) Can FCFS simulate priority scheduling for all possible parameters of priority scheduling? (How or why not?)
   b) Can priority scheduling simulate FCFS for all possible parameters of FCFS? (How or why not?)
4. **Round-robin and shortest job first (SJF)**
   a) State the parameters and behavior of round robin
   b) Can round robin simulate SJF for all possible parameters of SJF? (How or why not?)
   c) Can SJF simulate round robin for all possible parameters of round robin? (How or why not?)

PART-II (Practice Numerical)

Question 1

Draw the graph and compute waiting time and turnaround time for the following processes using **FCFS**, **SJF** and **SRTF**. [10]

Process	Arrival Time	Burst Time
P0	0	7
P1	3	9
P2	6	5
P3	8	12
P4	10	6
P5	11	5
P6	12	3

Question 2

Draw the graph and compute turnaround time for the following processes using **RR** Scheduling algorithm. Consider a time slice of 4 sec. Every even number process performs I/O after every 4 sec of its running life. I/O takes 8 seconds. [15]

Process	Arrival Time	Burst Time
P1	0	4
P2	1	5
P3	4	11
P4	5	9
P5	7	6
P6	10	10
P7	12	7
P8	14	5

Question 3

Schedule the following processes using RR. The processes P1, P2 and P3 have arrived at time units 0, 1 and 2 respectively. The number inside the parenthesis indicates the time units for CPU and I/O Bursts. Assume a time quantum of 4 time units.

P1	CPU Burst (6)	I/O Burst (8)	CPU Burst (3)	I/O Burst (4)	CPU Burst (5)

P2	CPU Burst (7)	I/O Burst (5)	CPU Burst (2)

P3	CPU Burst (5)	I/O Burst (5)	CPU Burst (4)	I/O Burst (4)	CPU Burst (3)

Question 4

Draw the graph for the following processes using Multi-level Queue and Multi-level Feedback Queue Scheduling algorithm.

Q1 - RR - 20 sec
Q2 - RR - 50 sec
Q3 - FCFS

Process	Arrival Time	Burst Time
P1	10	100
P2	50	120
P3	60	180
P4	65	110
P5	80	70

Question 5

Describe Scheduling algorithm of the UNIX System-V Release 3 operating system by drawing and labeling the diagram. Also mention its limitations.

PART-III (Shell Commands)

Write down a program in C, that receives two integers as command line arguments. The main thread creates a child thread and pass the two numbers to it after packaging those two numbers in a `struct`. The main thread waits for the termination of the child thread. The child thread function computes the power (first number raised to the power of second number). If the first number is 2 and the second number is 10 the thread function should display the result as 1024. The thread function should use `pow()` library function of math library. After the child thread terminates the main thread enters in an infinite loop doing nothing. Compile this program and make an executable with the name of your rollno. Then perform following tasks:

Task1: Use `readelf(1), od(1), size(1)` commands to get different attributes of your executable program file. Compile your program file with `-g` option to `gcc` and run the commands again to see the differences. Compile your program file with `--static` option to `gcc` and run the commands again to see the differences. Note down your observations.

Task2: Execute the program, suspend its execution and then run it in the background. Execute the program in the background, and then bring it to the foreground. Mean while in another terminal keep checking various statistics of your process using `ps(1)` with `-u` and `-l` options. Keep a note of all this on paper.

Task3: Use `top(1)` command and see confirm statistics that you have noted down using `ps(1)` in above task. Try changing its `nice` value and see the behavior of your process. Using `top(1)` send different signals to your running process and see whether a core file is generated or not. If not see what you need to change. Keep a note of all this on paper.

Task4: Learn how can you change the `nice` value of a running process from shell using the `renice(1)` command. Also see how you can run your program with a nice value of something other than default from the shell using `nice(1)` command. Keep a note of your observations on paper.

Task5: Use `schedtool(1)`, (install using `sudo apt-get install schedtool` command) to change following scheduling parameters of your running process and note you're your observations:

- Change scheduling policy SCHED_BATCH, SCHED_IDLEPRIO, SCHED_NORMAL
- Change nice value
- Change static priority
- Change cpu affinity