Birla Institute of Technology & Science - Pilani, Hyderabad Campus
CS F372 : Operating Systems
Mid-Sem Test

Type: Closed Book **Time:** 90 mins **Date:** Mar 12th, 2019 **Max Marks:** 50 (25% weightage)

Please Notes:

- You must answer all the parts of a question together as only the first occurrence of the answer to a question will be graded.
- Be cognizant of time. Be concise. Show work when needed. Write your answers legibly.
- Assume suitable data if necessary. Don't cripple the question with unnecessary assumptions.

---

1. You must answer the follwing questions sequencially and together. **[1 × 10 Mark(s)]**

   (a) What characteristic is common to traps, interrupts, and supervisor calls?

   (b) What kind of scheduling algorithm should be used for jobs such as computing pi to a million decimal places, computing the value of sine, cosine to a million decimal places, and finding the fourier transform of a fucntion?

   (c) What is the usual mode of communication in a monolitic kernel?

   (d) What is the minimum number of processes that can be in the ready state on a system with n CPUs and $2 \times n$ processes?

   (e) [**Yes or No**] Are the processes shown in the resource graph deadlocked (Fig 1)? (Assume all the standard notations).

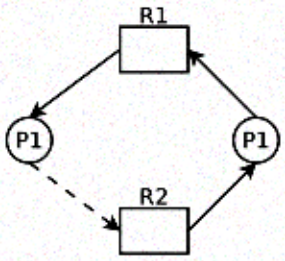

Figure 1: Simple RAG

   (f) [**True or False**]: When designing a multithreaded application, you must use synchronization primitives to make sure that the threads do not overwrite each other's TLB.

   (g) [**True or False**]: In general is the transit form thread state BLOCKED to thread state RUNNING legal (i.e. allowed).

   (h) At a particular time of computation, the value of a counting semaphore is 7. Then 20 wait operations and 'x' signal operations were completed on this semaphore. If the final value of the semaphore is 5, what is the value of 'x'?

   (i) Preemptive version of which scheduling is essentially Round Robin scheduling?

(j) Using Priority Scheduling algorithm, find the average waiting time for the following set of processes given with their properities in the order: **Process Name** : **Burst Time** : **Priority** respectively.

P1 : 10 : 3, P2 : 1 : 1, P3 : 2 : 4, P4 : 1 : 5, P5 : 5 : 2

2. (a) [**Agree or contradict with proper explanation**]: The operations on Input/Output Devices are synchronous. **[2 Mark(s)]**

(b) Critically explan the behaviour (not the console output) of the following code? Assume that all the necessary header files are included. **[3 Mark(s)]**

```
int main(){
        int c=0;
        while (fork() > 0) {
                printf("Hi\t"); c++;
        }
        printf("%d\t",c);
        return 0;
}
```

(c) Draw the process tree for the above program. Assume 1112 is the PID given to the above program, when it starts execution. **[2 Mark(s)]**

(d) Is the following solution provides controlled access to the critical section for two processes P_0 and P_1? Explain your answer pricesely. **[3 Mark(s)]**

```
do {
        flag[i] = true;turn = j;
        while (flag[j] && turn != j);
        //Critical section
        flag[i] = false;
        //Remainder section
} while (true);
```

3. (a) We have a system with multilevel feeb back queues, a CPU-bound process has a burst time of 40 seconds. This process does not have any I/O activity. If the first queue uses a time quanta of 2 seconds and at each level the time quantum increases by 5 seconds, how many times will the job be interrupted and what queue number will it be when it terminates? **[2 Mark(s)]**

(b) Enumerate the major differences between kernel-level threads and user-level threads along with suitable examples for each of them. (You must write in two column format.Only unique points are considered, while duplicates are ignored.) **[8 Mark(s)]**

4. (a) A certain program uses a thread pool, which is initially empty. When the program tells the thread pool to run a task in a separate thread, the thread pool reuses an existing idle thread if possible otherwise the thread pool creates and uses a new thread. Once a thread has finished running a task, the thread goes back into the thread pool as an idle thread. The program runs 50 tasks in sequence. Each task starts 50 milliseconds after the previous task started. The burst time of a task is 1 second. After all the tasks have finished, how many idle threads will be sitting in the thread pool? (Ignore any overhead apart from the task execution time.) Explain your answer. **[2 Mark(s)]**

   (b) Enumerate the challenges involved in various thread cancellation policies? **[2 Mark(s)]**

   (c) Enumerate the differences between downcall and upcall. (You must write in two column format.Only unique points are considered, while duplicates are ignored.) **[3 Mark(s)]**

   (d) Calculate the exponential averaging with T1 = 10, $\alpha = 0.5$ and the algorithm is SJF with previous runs as 8, 7, 4, 16. **[3 Mark(s)]**

5. (a) Which CPU scheduling policy that minimizes average completion time? What are the side effects of that CPU scheduling policiy? **[2 Mark(s)]**

   (b) For the processes listed in the Table 1, what is the average turnaround time, average wait time, and throughput using **Shortest Job First** and **Round Robin (Time Quantam =2)**.(Your answers should be upto two decimal places) **[8 Mark(s)]**

Table 1: Process Scheduling Data

Process	Arrival Time	CPU Burst
A	0.0	3
B	1.001	6
C	4.001	4
D	6.001	2

–End–