1. Briefly answer the following. [5*2=10]

a) **Explain how FCFS CPU scheduler penalizes I/O bound processes. [2]**

Ans: I/O bound processes have short CPU bust cycle. If there is a long CPU bound process, then I/O bound process has to wait to complete long CPU bust. In this way, FCFS penalizes I/O bound processes.

b) **Some simple OSs use non-preemptive scheduling. Give an example of a problem that can occur when the OSs uses non-preemptive scheduling. [2]**

Ans: When OS uses non-pre-emptive scheduling, the kernel process can not be interrupted. As a result the waiting process has to wait for the completion of kernel process. The execution time is unpredictable. As a result, it is difficult to guarantee application specific time constraints. Also, it is difficult to give CPU to high priority process soon after the interrupt.

c) **Discuss the concept of thread pools.[2]**

Ans: The concept of thread pools is used to increase efficiency of operating system. The idea is create a number of threads at system startup and place them in a pool, where they sit and wait for work. If the pool has no available thread, the server waits until one is free. It is faster to service with an exiting thread than waiting to create a new thread. A thread pool limits the number of threads at any point which is important on systems that can not support a large number of concurrent threads.

d) **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? [2]**

Ans: In multi-level scheduling, there are multiple queues; Each queue has a limit regarding time quantum. The time quantum increases as the queue level increases. A new process is assigned to first queue. If the burst time exceeds time quantum, it is pushed to the next level queue. A job at the lower queue will not be taken-up until the job queue of higher level queue is empty.

Strategy for computer use: The process should be programmed such that its burst time should be less than the time quantum period of the first queue.

e) **Discuss how the following pairs of scheduling criteria conflict in certain settings: CPU utilization and response time. [2]**

Answer: If the CPU utilization increases it does not mean that response time increases; on the other hand it may decrease. For example, if you are running a CPU bound job, the CPU utilization increases, but the response time of other processes are affected.

2. **In UNIX, I/O devices are treated as files. Discuss the advantages. [5]**

(Note: This question is from research paper)

Ans:
Each I/O device supported by UNIX is associated with at least one such file. Special files are read and written just like ordinary disk files, but requests to read or write result in activation of the associated device.

Advantage: There is a threefold advantage in treating I/O devices this way: file and device I/O are as similar as possible; file and device names have the same syntax and meaning, so that a program expecting a file name as a parameter can be passed a device name; finally, special files are subject to the same protection mechanism as regular files.

3. Explain the terms "CPU-bound process" and "I/O bound process" through examples. Suppose a CPU scheduling algorithm favors those programs that have used little processor time in the recent past. Explain why this algorithm favours I/O-bound processes and yet does not permanently deny processor time to CPU-bound processes? [5]

Ans: A CPU bound process uses significant CPU time. Example of CPU bound process is computation of fast fourier transform. An I/O bound process uses little processor time. Example of I/O bound process is file read.

Suppose ready queue contains both CPU bound and I/O bound processes. If the CPU gives priority to I/O bound process, it finishes I/O bound processes quickly and remaining time is used to process CPU-bound processes. In this way, both I/O utilization and CPU-utilization can be improved. Otherwise imagine a situation, if a processor gives priority to processes which consumes more CPU time in the past. In such a case I/O processes do not get CPU-time, if there are CPU-bound processes in the ready queue.

4.Consider a multiprocessor system and a multi threaded program using many-to-many threading model. Let the number of user-level threads in the program be more than the number of processors in the system. Discuss the performance implications of the following scenarios. [5]

- **(i) The number of kernel threads allocated to the program is less than the number of processors.**
- **(ii) The number of kernel threads allocated to the program is equal to the number of processors.**
- **(iii) The number of kernel threads allocated to the program is greater than the number of processors but less than the number of user-level threads.**

Ans:

- (i) All the processors/kernel threads are not fully utilized. Some of the kernel threads will not have any work. So, the system in underutilized.
- (ii) All the processors are utilized. However, if user program executes blocking system call through kernel thread, CPU will be idle.
- (iii) CPU utilization will be high (modern operating systems). Since the number of kernel threads are more than the number of processors, if one kernel thread is blocked, another kernel thread can be allocated to CPU. Similarly, for each kernel thread we can attach more than one user-level thread. In this case, if user-level thread makes a blocking system call, another user-level thread can be executed by kernel thread.

5.UNIX is unsuitable for real-time applications because a process executing in kernel mode may not be interrupted. Explain the reasons. [5]

6.A system is receiving both batch jobs and interactive jobs. Normally, round robin makes the users of interactive jobs happy. Consider a variation of round robin we will call progressive round robin. In progressive round-robin, each process has its own time quantum. This starts out at 50ms, and increases by 50 ms each time it goes through the round-robin queue. Give the advantages and disadvantages of this variant over ordinary-round robin with respect to the users of batch jobs and interactive jobs. [5]

Ans

Ordinary round robin: Independent of burst time, the ordinary round robin gives equal priority to all kinds of jobs. As a result, the interactive processes which have burst time less than time quantum are processed with good response time. A process is guaranteed to get the CPU slot after (time quantum*(MPL-1)). The interactive processes

Code Implementation:

```
monitor bridge{
   semaphore num_waiting_north=0;
   semaphore num_waiting_south=0;
   int on_bridge=0;
   condition free=0;
   int prev=0;

      void enterbridge_north(){
            num_waiting_north++;
            while(on_brdige || (prev==0 && num_wait_south>0))
               free.wait();
               num_waiting_north--;
            prev=0;
            }
            }
            void exit_bridge_north(){
               on_bridge=0;
               free.broadcast();
            }

            void enter_bridge_south(){
```