INDIAN INSTITUTE OF TECHNOLOGY, KHARAGPUR

Date ....FN / AN *Time:* 2/3 Hrs. *Full Marks* .... *No. of Students* ....
Autumn / Spring Semester, 20 .... Deptt. .... Sub No. ....
......Yr. B. Tech.(Hons.) / B. Arch. / M. Sc. Sub. Name ....

Instructions : **Answer All (4) Questions.**

1. [$\mathbf{10 \times \frac{1}{2}}$]

(a) Which of the following is not a function of an operating system?
   - i. Generating interrupts
   - ii. Making the computer system covenient to use
   - iii. Manage I/O devices
   - iv. Protecting user programs from one another

(b) Which of the following might use spooling?
   - i. Protecting jobs from writing into the wrong memory location.
   - ii. Doing long term job scheduling.
   - iii. Reading jobs from cards onto the disk and loading new jobs from disk into empty memory partitions.

(c) Choose the best definition for Process:
   - i. An executable program
   - ii. Program code + contents of processor's registers + stack
   - iii. Program code + contents of processor's registers + stack + PCB + ready queue
   - iv. Program code + contents of processor's registers

(d) What is the main advantage of multiprogramming?
   - i. Efficient use of the CPU
   - ii. Fast response
   - iii. Efficient use of disk

(e) What is the difference between multithreading and multiprocessing?
   - i. Multiple threads can share code and the data section.
   - ii. Only processes require context switching.
   - iii. Only threads can support parallelism.

(f) Which component ensures that a process can execute only within its own address space?
   - i. I/O Device
   - ii. Memory-addressing hardware
   - iii. Stack pointers

Sig.of the Paper-Setter ....

(g) Which of the following instructions should be privileged?

i. Read data from disk

ii. Set priority of a process

iii. Read the clock

(h) Thread of the same task... I. Share the same address space II. Reduce context switching overhead. III. Are protected from each other the same weight as heavy-weight processes.

i. Only I

ii. I and II

iii. I, II, and III

(i) Assume a single threaded kernel OS running multiple user threads. If one user thread requests a read system call then...

i. other system threads continue to run.

ii. some user threads run and some are blocked.

iii. all user threads are blocked.

(j) Which of the following statements about process state transitions is FALSE?

i. When a running process receives an interrupt it goes to the ready state.

ii. Upon finishing execution the running process exits and goes to the terminated state.

iii. A ready process goes to the running state when the scheduler decides to schedule it.

iv. A waiting process upon completion of I/O goes back to the running state.

2. $[\mathbf{5} \times \mathbf{1}]$

Give brief answers to the following questions.

(a) Why is it necessary to keep the relocation information in a *object module*?

(b) Does the OS need support from architecture to protect one process from another?

(c) In what situation does the OS go for memory compaction?

(d) In what situation two processes share the same code? How does paged memory management help in this sharing?

(e) What is the roll of a *base* and a *limit* registers in program loading and protection? How are these registers protected from a user process?

3. $[(\mathbf{3} + \mathbf{2} + \mathbf{1}) + \mathbf{4}]$

(a) Consider the workload in the following table:

Process	Burst Time	Priority	Arrival Time
P1	10	4	1
P2	12	3	0
P3	5	2	2
P4	8	1	4

Sig.of the Paper-Setter ....

Draw the Gantt chart for the preemptive shortest job first and preemptive priority scheduling. What is the waiting time for each process for each of the algorithm above? What is the average waiting time for each?

(b) Fix the following code sample considering the three criteria for synchronisation.

```
/*** Producer Consumer Using Semaphores **/
semaphore mutex = 1;
semaphore empty = N;
semaphore full = 0;
int buffer[N];

void producer() {
int item;
while(TRUE) {
item = produce_item();
wait(mutex);
wait(empty);
insert_item(item,buffer);
signal(full);
signal(mutex);
}
}

void consumer() {
int item;
while(TRUE) {
wait(mutex);
wait(full);
item = remove_item(buffer);
signal(empty);
signal(mutex);
}
}
/** Producer Consumer Using Semaphores **/
```

4. $[\mathbf{4+2+3+1}]$

A CPU generates 43-bit virtual address. The main memory is divided in 8KByte page frames.

(a) Give a 3-level page table scheme for virtual to physical address translation.

(b) Clearly specify the important fields and their sizes in the entries of different tables.

(c) How many of such tables will be used for a process with a logical address space **0x2 00 0000 0000** to **0x2 00 0002 FFFF**?

(d) What is the main disadvantage of this 3-level translation scheme?

Sig.of the Paper-Setter ....