INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD

Mid-Semester Examination, September 2016

Date of Examination (Meeting) : 30.09.2016 (1st Meeting)

Program Code & Semester : B.Tech.(IT)/ B.Tech.(ECE)/ Dual Degree - 3rd Semester

Paper Title: Operating System
Paper Code: IOPS332C

Paper Setter: Dr. Bibhas Ghoshal
(Sec. A - BG, Sec. B – JS/AD, Sec. C - BG, RGIIT - SA)

Max Marks: 30 **Duration: 2 hours**

Note: There are **six questions** in this question paper. **Answer All questions**. The subquestions of Question 1 (multiple choice questions) are worth 0.5 mark each with a *penalty of 0.25 marks for incorrect answer*, the marks for each sub question in Question 2 (multiple choice questions) are worth 2 mark each. However, for each sub question in Question 2 you need to **justify your answer**, failing which your answer will not be considered. The marks for rest of the questions have been provided alongside each question.

Q1. Choose the best option from the following (**no justification needed**). [5 marks]

(A). Which of the following is not a function of the operating system?
(i) Generate interrupts
(ii) making the computer sytem convenient to use
(iii) manage i/o devices
(iv) protecting user programs from one another

(B). Choose the best definition of 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

(C). What is the difference between multithreading and multiprocessing?
(i) multiple threads share code and data sections
(ii) only processes require context switching
(iii) only threads can support parallelism

(D). What is the advantage of multiprogramming:
(i) efficient use of CPU
(ii) fast response
(iii) efficent use of disk

(E). Which component ensures that process can execute within its own address space?
(i) I/O device
(ii) memory addressing hardware
(ii) stack pointers

(F). Which of the following instructions should be priviledged?
(i) Read data from disk
(ii) set priority of process
(iii) read the clock

(G). Threads of the same task....I. Share the same address space II. Reduce context switching overhead III. Are protected from each other the same way as heavy weight processes. Which of the following options are correct ?
(i) Only Statement I about threads is true
(ii) Statements I and II about threads are both true
(iii) Statements I, II and III about threads are all true

(H). Assume a single thread 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

(I). Which of the following statements about the process state transitions are FALSE:

(i) When a running process receives interrupt, it goes to ready state.
(ii) Upon finishing, the running process exits and goes to terminated state.
(iii) A ready state goes to running state when the scheduler schedules it.
(iv) An I/O process on completion of I/O request goes back to running state.

(J) Pipe is used for interprocess communication. Which statements about PIPE are true?

(i) we may read and write from pipe at the same time
(ii) pipe is used for unidirectional flow of data
(iii) The pipe() system call requires an array of two integers as parameter.
(iv) All of these

2. Choose the best option with justification in each case. Without justification you would not be awarded any marks for your answer. [10 marks]

(A). Consider the following code fragment:

```
    if (fork() == 0)
            { a = a + 5; printf(“%d,%d\n”, a, &a); }
    else { a = a – 5; printf(“%d,%d\n”, a, &a); }
```

Let u, v be the values printed by the parent process, and x, y be the values printed by the child process. Which one of the following is TRUE? Why?
(i) u = x + 10 and v = y
(ii) u = x + 10 and v != y
(iii) u + 10 = x and v = y
(iv) u + 10 = x and v != y

(B). Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared boolean variables S1 and S2 are randomly assigned.

Method used by P1	Method by P2

While (S1==S2);
    critical section
S1=S2;
	While (S1!=S2);
	    critical section
	S2 = not(S1);

Which one of the following statements describes the properties achieved?
(i) Mutual exclusion but not progress
(ii) Progress but not mutual exclusion
(iii) Neither mutual exclusion nor progress
(iv) Both mutual exclusion and progress

(C). The enter_CS() and leave_CS() functions to implement critical section of a process are realized using test-and-set instruction as follows:

```
void enter_CS(X)
{
        while test-and-set(X) ;
}

void leave_CS(X)
{
        X = 0;
}
```

In the above solution, X is a memory location associated with the CS and is initialized to 0.
Now consider the following statements:
I. The above solution to CS problem is deadlock-free
II. The solution is starvation free.
III. The processes enter CS in FIFO order.
IV. More than one process can enter CS at the same time.

Which of the above statements is TRUE?
(i) I only
(ii) I and II
(iii) II and III
(iv) IV only

(D). A uni-processor computer system only has two processes, both of which alternate 10 ms CPU bursts with 90 ms I/O bursts. Both the processes were created at nearly the same time. The I/O of both processes can proceed in parallel. Which ofthe following scheduling strategies will result in the least CPU utilization (over a long period of time) for this system?
(i) First come first served scheduling
(ii) Shortest remaining time first scheduling
(iii) Static priority scheduling with different priorities for the two processes
(iv) Round robin scheduling with a time quantum of 5 ms

(E). The following program:

```
main(){
        if(fork()>0)
        sleep(100);
}
```

results in the creation of:
(i) an orphan process
(ii) a zombie process
(iii) a process that executes forever
(iv) None of these

3. (A). You write a UNIX shell, but instead of calling fork() then exec() to launch a new job, you instead insert a subtle difference: the code first calls exec() and then calls fork() like the following:

[2 marks]

```
shell (..) {
.. ..
exec (cmd, args);
fork();
.. ..
}
```

Does it work? What is the impact of this change to the shell, if any? Explain.

3. (B). What is busy waiting with respect to a critical section problem? Can busy waiting be avoided? [2 marks]

4. 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

Draw the Gannt chart for preemptive shortest job first and preemptive priority scheduling . What is the average waiting time and response time in each case? [4 marks]

5 (A). What are user level and kernel level threads? [1mark]

(B). Assume you want to implement a web-server for YouTube by using multithreading, where each thread serves one incoming request by loading a video file from the disk. Assume the OS only provides the normal blocking read system call for disk reads. Which threads should be used, user-level threads or kernel-level threads? Why? [2 marks]

OR

(B). You want to implement a web-server for Facebook, to serve each user's "Home" page (the first page you see after you log in). Your web-server needs to perform many tasks: load the news feeds from each of your friends, load the advertisement, check for new messages, etc. Now you want to implement your web-server by using multithreading, and have one thread to perform each of the tasks, and later these threads will cooperate with each other to collectively construct the "Home" page. For performance reasons, Facebook makes sure that all the data these threads need is already cached in the memory (so they don't have to perform any disk I/O). What do you use, user-level threads or kernel-level threads? Why?

[2 marks]

6 (A). List the different inter process communication (IPC) mechanisms (with examples).
[2 marks]

(B). The following code in C uses an IPC mechanism to communicate between two processes. Identify the IPC mechanism used and document the code (fill the comments sections in the code) to highlight the usage of the different POSIX system calls used for IPC. Additionally, you need to comment on the output. [2 marks]

```
main(){
        int shmid,status;
        int *a, *b;
        int i;

      shmid = shmget(IPC_PRIVATE, 2*sizeof(int), 0777|IPC_CREAT);

           /*--------Comments----- */

     if (fork() == 0) {                          /*--------Comments----- */

          b = (int *) shmat(shmid, 0, 0);        /*--------Comments */

           for( i=0; i< 10; i++) {

                      sleep(5);

                    printf("\t\t\t Child reads: %d,%d\n",b[0],b[1]); /*----Comments---- */

                }

        shmdt(b);                                /*----Comments---- */

       }

        else {                                   /*----Comments---- */

          a = (int *) shmat(shmid, 0, 0);        /*----Comments----- */

          a[0] = 0; a[1] = 1;

               for( i=0; i< 10; i++) {
                       sleep(1);
                       a[0] = a[0] + a[1];
                       a[1] = a[0] + a[1];
                       printf("Parent writes: %d,%d\n",a[0],a[1]);  /*----Comments---- */
               }

              wait(&status);                     /*-----Comments----- */

             shmdt(a);                           /*----Comments----- */

              shmctl(shmid, IPC_RMID, 0);        /*----Comments------ */

 }}
```
