Name:_____ **Enroll:_____**

Jaypee Institute of Information Technology

Test - 1 Examination, October 2021-22 Odd Semester

B.Tech, Semester-V

Course Title: Operating Systems and Systems Programming **Max. Time: 1 hour**
Course Code: 15B11CI412 **Max Marks: 20**

Section - I: Answer the questions briefly with proper justification: 10 X 1 Mark each.

Q1. [C311.1] Which of the following instructions will execute in Kernel mode?

I. Changing the value of the time setting in the system
II. Changing the value of memory management information
III. Changing the value of a variable in user program
IV. Change processor mode from Kernel to user

Ans: I, II, IV (Strict marking either 1 or 0)

Q2. [C311.2] Consider you are designing a chat application and you are creating a new process for each user in the system. What type of scheduling algorithms should be uses to schedule these processes?

Ans: System should use **preemptive scheduling** to provide quick service response to interactive and I/O bound processes. **(Strict marking either 1 or 0)**

Q3. [C311.3] Complete the program by filling missing entries.

```
void *BW(void *n)
{
        int i;
        long tid;
        double r=0.0;
        tid = (long)n;
        printf("Thread %ld starting...\n",tid);
        for (i=0; i<1000000; i++)
                r = r + sin(i) * tan(i);
        printf("Thread %ld done. Result = %e\n",tid, r);
      pthread_exit((void*) n);
}
int main (int argc, char *argv[])
{
pthread_t thread[4];
pthread_attr_t attr;
  int receive;
  long i;
```

```
  void *status;
pthread_attr_init(_____); // Statement-1
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
for(i=0; i<4; i++)
{
        printf("Main: creating thread %ld\n", i);
    receive = pthread_create(_____, _____, _____, (void *)i); //Statement-2
    if (receive)
   { printf("ERROR; return code from pthread_create() is %d\n", receive); exit(-1);  }
 }
pthread_attr_destroy(_____); //Statement-3
for(i=0; i<4; i++)
{     receive= pthread_join(_____, &status); Statement-4
    if (receive) { printf("ERROR; return code from pthread_join() is %d\n", receive); exit(-1);}
    printf("Main: completed join with thread %ld having a status of%ld\n",i,(long)status);
 }
printf("Main: program completed. Exiting.\n");
pthread_exit(NULL);
}
```

Q3(A) [C311.3] Complete Statement 1 and 2 of the given program.

Q3(B) [C311.3] Complete Statement 3 and 4 of the given program.

Solution: (Strict marking either 0.5 or 0 for each correct ans)

pthread_attr_init(&attr);

receive = pthread_create(&thread[i], &attr, BW, (void *)i);

pthread_attr_destroy(&attr);

receive= pthread_join(thread[i], &status);

Q4. [C311.2] Suppose there are two threads executing simultaneously. One is printing numbers from 1 to 1000000 and another is printing strings with numbers as characters from 1 to 1000000. Which thread is going to complete its execution first.

Ans: 2nd thread completes its execution first **(Strict marking either 1 or 0)**

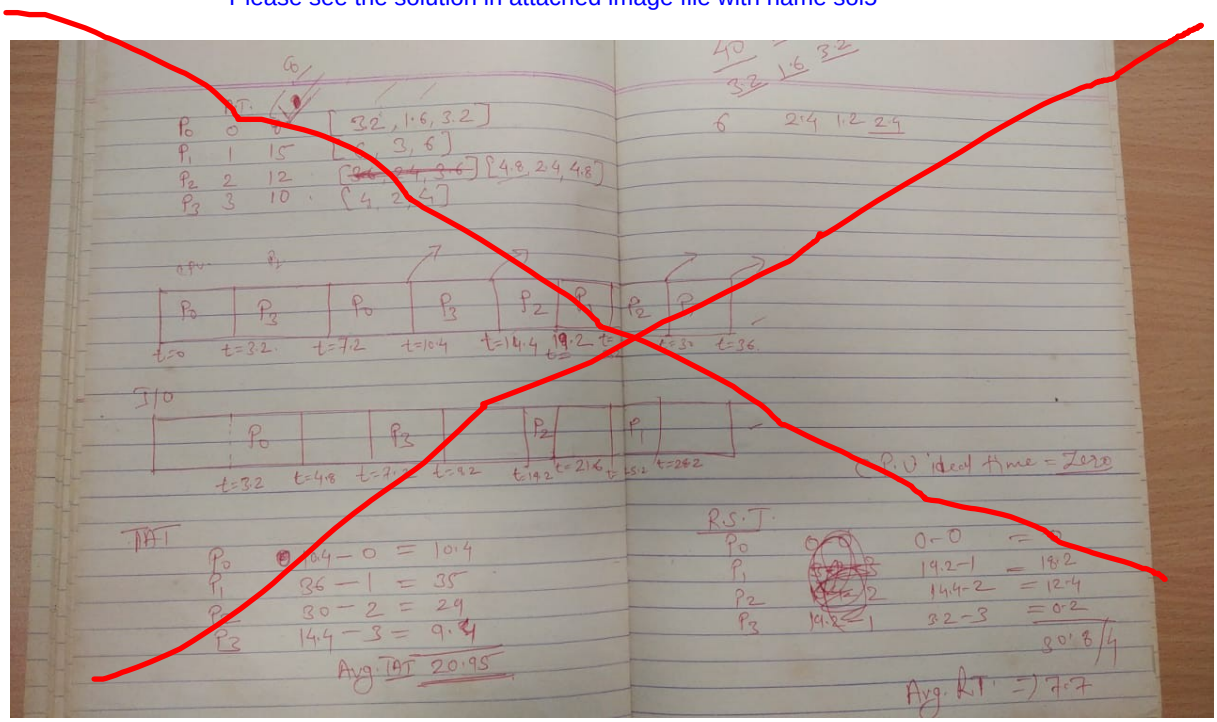
Q5. [C311.2] Suppose four processes are arriving at times 0, 1, 2 and 3 with their total burst time of 8, 15, 12 and 10 units, respectively. Each process spends the first 40% of their burst time in doing CPU computations, the next 20% in doing I/O activity, and the last 40% of time in doing CPU computation again. The scheduler uses a shortest job first scheduling algorithm for CPU computations and preemptive shortest job first scheduling algorithm for performing I/O activity. Assume that all processes are wait in a single queue for I/O operations.

Q5(A) Prepare Gantt Charts. **(Strict marking either 1 or 0)**

Q5(B) Calculate average turnaround time and response time for each process. **(Strict marking either 0.5 or 0 for each component)**

Q5(C) Calculate the percentage of time does the CPU and I/O devices remain idle? **(Strict marking either 1 or 0)**

Please see the solution in attached image file with name sol5



Q6. [C311.4] Consider the following critical section (C.S.) problem with four processes P_i, P_j, P_k and P_m. Here, 'T' represents 'true', 'F' represents 'false' and 'C.S.' represents critical section. The 'flag' is a Boolean array initialized by F (False).

```
Pi:
 Repeat
        flag[i] := T;
        while flag [j] or flag[k] or flag[m]  do skip;
        Switchcase (turn)
              j: if flag [j] then
                 begin
                       flag [i] := F;
                       while turn != i do skip;
                       flag [i] := T;
                 end;
              k: if flag [k] then
                 begin
                      flag [i] := F,
                      while turn != i do skip;
                      flag [i] := T
                 end
              m: if flag [m] then
                  begin
                      flag [i] := F,
                      while turn != i do skip;
                      flag [i] := T
                  end
```

```
        C.S.
        if turn = i then turn := j; //Statement: S1
        flag [i]:= F // Statement: S2
        //non-critical section
until false;
```

Q6(A) Does the given solution ensures mutual exclusion, progress and bounded waiting in the critical section if turn is initialized by 'j'? Give reason in one line.

Ans: Mutual Exclusion and Bounded wait is guaranteed but Progress is not guaranteed. **(Strict marking either 1 or 0)**

Q6(B) What is the role of statements S1 and S2 in this solution to satisfy the requirements of a critical section solution?

Ans: Bounded Waiting is guaranteed as strict alteration (for 4processes) is being used by this synchronization construct by setting turn variable appropriately in the exit section before the control passes to other process. **(Strict marking either 1 or 0)**

Section II: Solve the following Questions with proper steps.

Note: Preparing appropriate Gantt Chart is must for questions on CPU scheduling.

Q7. [C311.2] [2-Marks] For the given processes and their corresponding burst time, what should be the size of time quantum in Round Robin scheduling to achieve optimal throughput but it does not turn into a First Come First Serve scheduling? Explain and prove by solving. Assume context switch overhead is 2 units of time.

Process	Burst Time
P1	7
P2	5
P3	8
P4	4

Ans: 1 mark for theory and formula(flexible marking), 1 mark for calculations and gantt chart

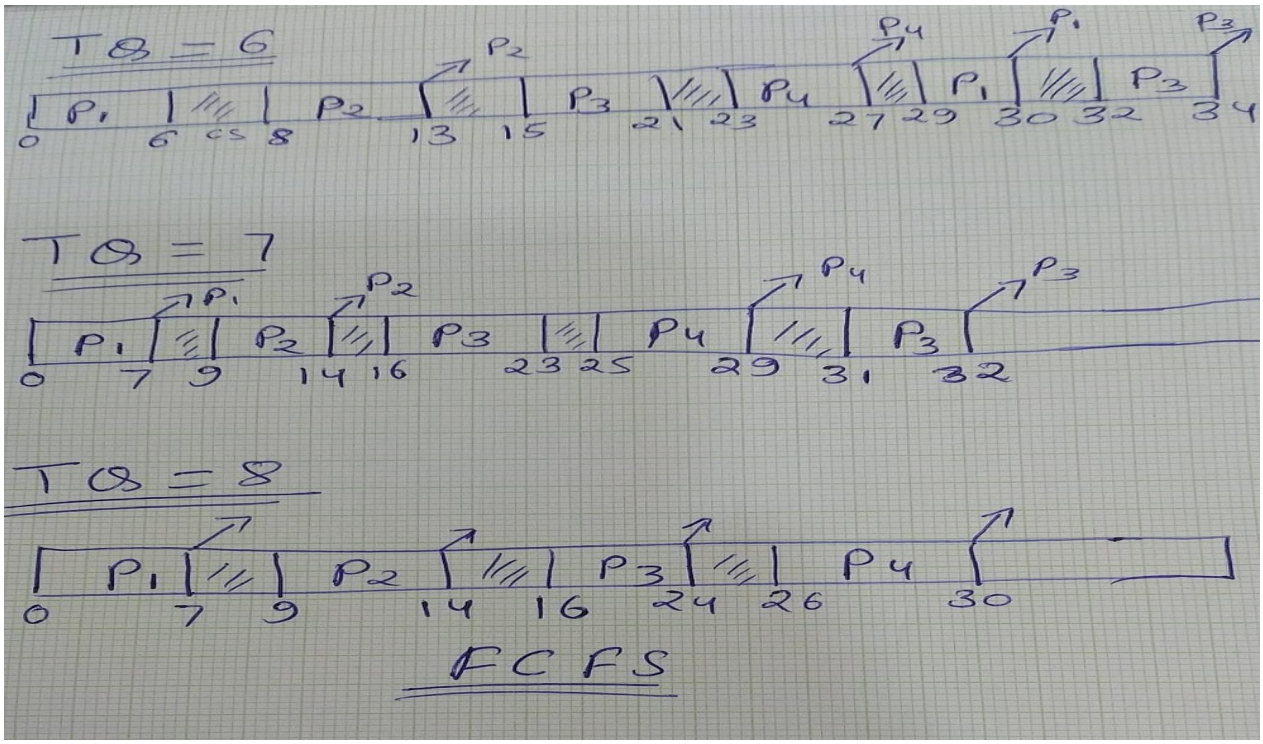
Time quantum should be Greater than the average process burst time.

Throughput = no of processes / (max of completion time - time at which first process is brought to CPU)

With TQ<7 processes completion time is 34 and hence throughput is 4/34.

With TQ=7 processes completion time is 32 and throughput is 4/32.

In the given scenario, TQ=8 acheives maximum throughput with processes completion time as 30 and throughput 4/30. But it is equivalent to FCFS algorithm.



Q8. [C311.2] [3-Marks] Write a C program to create three processes (a parent, a child and a grandchild) which communicate with each other using ordinary pipes, such that

a. Process 1 (Parent) displays a "Hello User" message along with its unique process id and asks the user to provide a string. It reads the string input provided by the user and sends it to the process 2.

b. Process 2 (Child) counts the number of vowels and consonants in the received string and displays the count to user along with its unique process id. It also sends the received vowels to the process 3.

c. Process 3 (Grandchild) displays the received vowels along with its unique process id to the user.

Note: Attaching output screenshot is mandatory

Solution:

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<string.h>

#include<sys/types.h>

int main()

{
```

```
//0.5 marks for creating pipes
int fd1[2], fd2[2];
int pid, pid1, i=0, v=0,c=0;
char recv[100], str[100], *strp, strv[100], Vrecv[100];
pipe(fd1);
pipe(fd2);
pid=fork();
//0.5 mark for grandchild code
if(pid==0)
{
   pid1=fork();

   if(pid1==0)
   {
      close(fd2[1]);
      FILE *in = fdopen(fd2[0], "r");
      fscanf(in, "%s", Vrecv);
      printf("Grand child with pid: %d received vovels= %s \n", getpid(), Vrecv);

   }
//1 mark for following child code
   else if(pid1>0)
   {
   close(fd1[1]);
         FILE *in = fdopen(fd1[0], "r");
         fscanf(in, "%s", recv);
   printf("Child with pid= %d received string= %s \n", getpid(),recv);
   strp=recv;
   while(*strp!='\0')
   {
      if(*strp=='A' ||*strp=='E' ||*strp=='I' ||*strp=='O' ||*strp=='U'||*strp=='a' ||*strp=='e' ||*strp=='i' ||
*strp=='o' ||*strp=='u')
      {v++;
         strv[i]= *strp;
```

```
            i++;
        }
        else
            {c++;}
         strp++;
    }
        strv[i]= '\0';
        printf("Vowels are: %s\n",strv);
        printf("No. of consonants are %d \n ",c);

        close(fd2[0]);
        FILE *out = fdopen(fd2[1], "w");
        fprintf(out, "%s", strv);

    }
    else{printf("grandchild not created");}
}
else
{  //0.5 mark for following parent code
    close(fd1[0]);
    printf("Enter a string in parent with pid= %d: ", getpid());
    fgets(str, 100, stdin);
            FILE *out = fdopen(fd1[1], "w");
            fprintf(out, "%s", str);

}
return 0;
}
```

Output with Linux compiler and Online compiler is given (.5 marks for any one correct output attached)

```
shilpa@shilpabudhkar:~/Dropbox/ODD 2021/SDF1$ ./prog
Enter a string in parent with pid= 25777: gjsgd,fhdgjlghfkjhdv
Child with pid= 25778 received string= gjsgd,fhdgjlghfkjhdv
Vowels are:
No. of consonants are 20
 Grand child with pid: 25779 received vovels=
shilpa@shilpabudhkar:~/Dropbox/ODD 2021/SDF1$ ./prog
Enter a string in parent with pid= 25786: sfjkajfgersddbcvaewiuryiurti
Child with pid= 25787 received string= sfjkajfgersddbcvaewiuryiurti
Vowels are: aeaeiuiui
No. of consonants are 19
 Grand child with pid: 25788 received vovels= aeaeiuiui
shilpa@shilpabudhkar:~/Dropbox/ODD 2021/SDF1$
```

Programiz
C Online Compiler

Learn C App

main.c

Run

Output

Clear

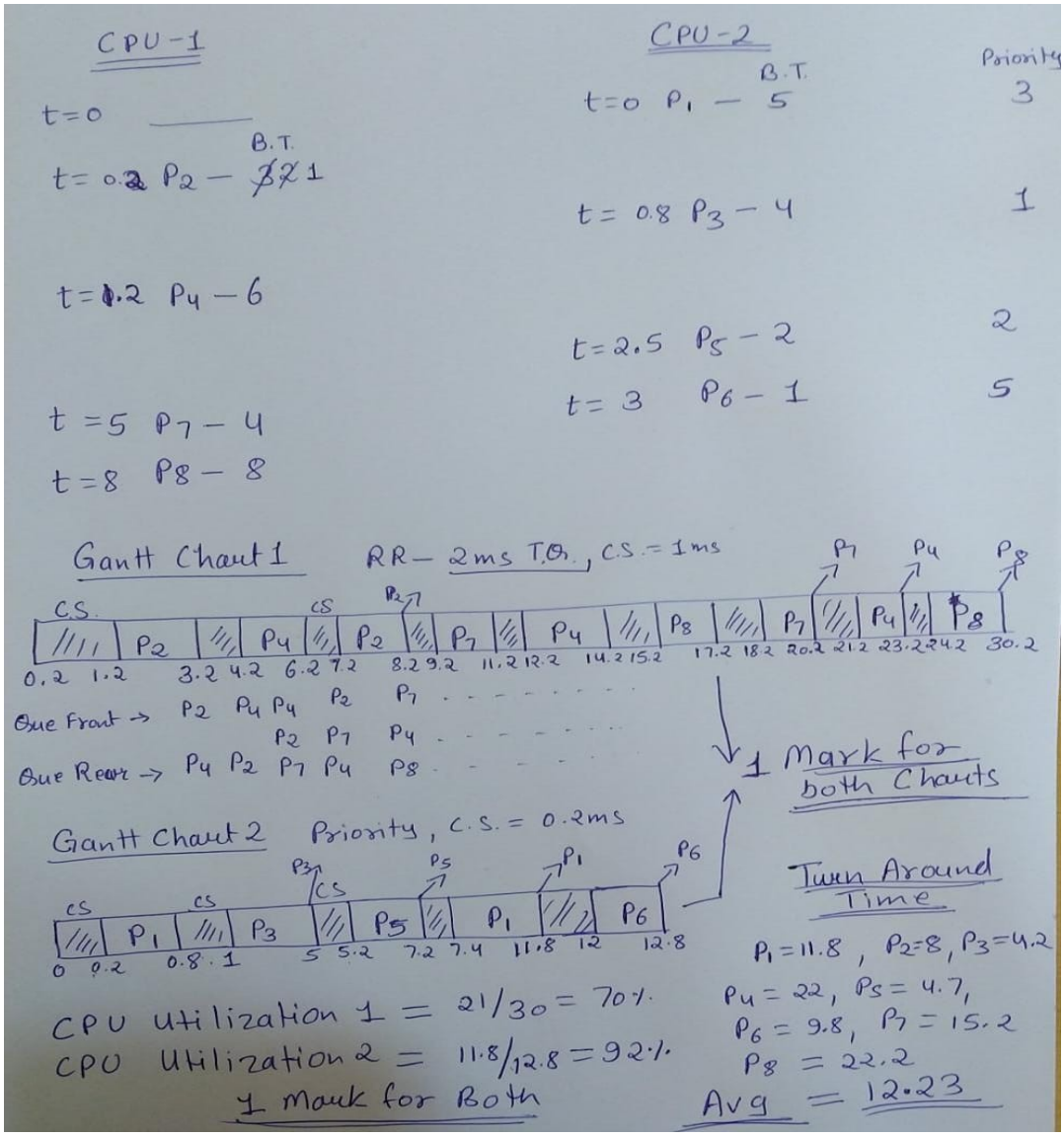
```c
// Online C compiler to run C program online
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<string.h>
#include<sys/types.h>
int main()
{
int fd1[2], fd2[2];
int pid, pid1, i=0, v=0,c=0;
char recv[100], str[100], *strp, strv[100], Vrecv[100];
pipe(fd1);
pipe(fd2);
pid=fork();

if(pid==0)
{
    pid1=fork();
```

```
/tmp/Q4qwiAsLsM.o
Enter a string in parent with pid= 1884: asdefg
Child with pid= 1885 received string= asdefg
Vowels are: ae
No. of consonants are 4
 Grand child with pid: 1886 received vovels= ae
```

Q9. [C311.2] [3-Marks] Consider eight soft real time processes are executed in a multiprocessor system with two processors using Linux platform. Assume multiprocessor scheduling with one global ready queue and per processor local ready queue. Processes are first stored in a global ready queue, and then dispatched to local ready queue for execution in one of the processors based on their processor affinity. Operating system schedules processes using round robin scheduling algorithm for execution in processor-1 with time quantum equal to 2ms and using pre-emptive priority scheduling for execution in processor-2. Assume the context switching time of the processes is 1ms for 1st processor and 0.2 ms for 2nd processor. The following table shows the CPU burst time, arrival time, priority and processor affinity of the processes.

Process	Priority	Arrival Time	CPU Burst	Processor Affinity
P1	3	0.0	5	2
P2	1	0.2	3	1
P3	1	0.8	4	2
P4	4	1.2	6	1
P5	2	2.5	2	2
P6	5	3	0.8	2
P7	7	5	4	1
P8	8	8	8	1

Prepare Gantt chart for both the processors. Calculate CPU utilization of both the processors and average turn around time of all the processes.



Solution: Strict marking, 1 mark-Gantt chart for both the processors. 1 mark- CPU utilization of both the processors and 1 mark - average turn around time of all the processes.

Q10. [C311.4] [2 marks] Suppose the following process consists of three threads.

Process: int x=y=1;

t1:	t2:	t3:
x=2;	j=x;	k=y;
i=x+1;	x=2*j;	y=k*x;

a) Does this process consists of any critical section? If yes, mention the statement.
b) Does it result in any race condition? Explain.

Ans:

a) Every statement of every thread is accessing a shared variable. Hence, every statement comes under critical section. (1 mark, flexible marking)

b) The final value of the processes computation depends on the interleaving of concurrent threads statements. Hence, there is Race Condition. (1 mark, flexible marking)