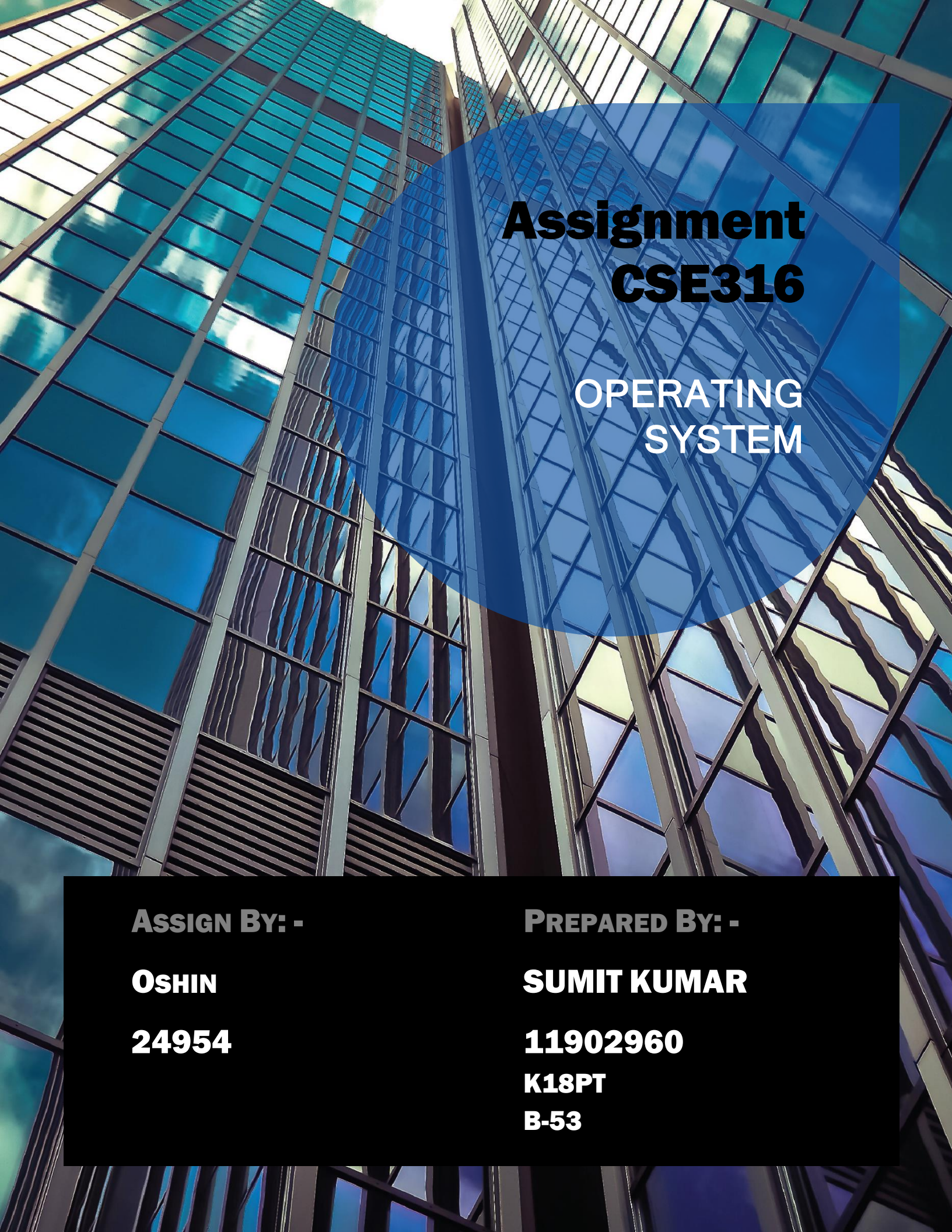

Assignment CSE316

OPERATING SYSTEM

ASSIGN BY: -

OSHIN

24954

PREPARED BY: -

SUMIT KUMAR

11902960

K18PT

B-53

Simulation Based Assignment

Question:-

Q9).

Design a scheduler that uses a preemptive priority scheduling algorithm based on dynamically changing priority. Larger number for priority indicates higher priority. Assume that the following processes with arrival time and service time wants to execute (for reference):

Process ID	Arrival Time	Service Time
P1	0	4
P2	1	1
P3	2	2
P4	3	1

When the process starts execution (i.e. CPU assigned), priority for that process changes at the rate of m=1.When the process waits for CPU in the ready queue (but not yet started execution), its priority changes at a rate n=2. All the processes are initially assigned priority value of 0 when they enter ready queue for the first time. The time slice for each process is q = 1. When two processes want to join ready queue simultaneously, the process which has not executed recently is given priority. Calculate the average waiting time for each process. The program must be generic i.e. number of processes, their burst time and arrival time must be entered by user.

Solution:-

C-Program Written Code: -

```
#include<stdio.h>
#include<conio.h>
struct process
{
    int processID;
    int burstTime;
    int arrivalTime;
    int priority;
    int waitTime;
```

```
};
//to calculate whole complete completion time and burst time
int total_time,burst_time=0;
int total= -1, i= -1;
/*to make buffer of queue in which process will be entered and final process
after completion to be entered in result of buffer 100;*/
struct process queue[100],result[100],swap;
//Asking Number of process to be created.
int process_create()
{
int n;
clrscr();
printf("Enter the number of process you want to be processed:");
scanf("%d",&n);
return n;
}
//To Execute the process and Finish.
void execute()
{
if(total>=0)
{
int wait,j;
//to increase the priority and decreaseBurst Time.
if(burst_time!=0 && queue[0].burstTime!=0)
{
queue[0].burstTime--;
burst_time--;
queue[0].priority++;
queue[0].arrivalTime=total_time+1;
total_time++;
//to increase the waiting time and priority of wait processes.
```

```
for(wait=1;wait<=total;wait++)
{
queue[wait].priority+=2;
queue[wait].waitTime=++queue[wait].waitTime;
}
}
//Storing Completed Processes In Result Queue.
if(queue[0].burstTime==0)
{
i++;
result[i]=queue[0];
for(wait=0;wait<total;wait++)
{
queue[wait]=queue[wait+1];
}
total--;
}
//Sorting Processes According To Their Priority
for(wait=0;wait<total;wait++)
{
for(j=0;j<total;j++)
{
if(queue[wait].priority<=queue[j].priority)
{
swap=queue[wait];
queue[wait]=queue[j];
queue[j]=swap;
}
}
}
if(queue[0].priority<=queue[1].priority && total>=1)
```

```
{
swap=queue[0];
for(wait=0;wait<total;wait++)
{
queue[wait]=queue[wait+1];
}
queue[total]=swap;
}
}
}

/*In Main Function Taking Arrival Time, Burst Time From User For The No Of Processes Entered
By Them and sorting the array according to arrival time and if arrival time
is equal then sorting according to burst time here.*/
void main()
{
    int l,j,n=process_create(),count=0;
    float averageWaitTime=0;
    struct process pcreate[4];
    clrscr();
    for(l=0;l<n;l++)
    {
    pcreate[l].processID=l+1;
    printf("\nEnter Arrival time of process[%d]: ",l+1);
    scanf("%d",&pcreate[l].arrivalTime);
    printf("\nEnter Burst time of process[%d]: ",l+1);
    scanf("%d",&pcreate[l].burstTime);
    pcreate[l].priority=0;
    pcreate[l].waitTime=0;
    burst_time=burst_time+pcreate[l].burstTime;
    }
```

```
    for(l=0;l<n;l++)
    {
    for(j=0;j<n;j++)
    {
    if(pcreate[l].arrivalTime<pcreate[j].arrivalTime)
    {
    swap=pcreate[l];
    pcreate[l]=pcreate[j];
    pcreate[j]=swap;
    }
    if(pcreate[l].arrivalTime==pcreate[j].arrivalTime)
    {
    if(pcreate[l].burstTime<=pcreate[j].burstTime)
    {
    swap=pcreate[l];
    pcreate[l]=pcreate[j];
    pcreate[j]=swap;
    }
    }
    }
}
//printing the sorted process id with respect to arrival time and if
//arrival time is equal than burst time.
    printf("\n.\n.\n.\nValues Entered are : (Table Sorted According To The Arrival Time)\n");
    printf("PROCESS TABLE \n");
    printf(".....\n");
    printf(" PROCESS ID          ARRIVAL TIME       SERVICE TIME \n");
    printf(".....\n");
    for(l=0;l<n;l++)
    {
    printf(" P%d                 %d
%d\n",pcreate[l].processID,pcreate[l].arrivalTime,pcreate[l].burstTime );
```

```
}
total_time=pcreate[0].arrivalTime;
for(j=pcreate[0].arrivalTime;j<=pcreate[n-1].arrivalTime;j++)
{
for(l=0;l<n;l++)
{
if(pcreate[l].arrivalTime==j && count!=n)
{
total++;
queue[total]=pcreate[l];
count++;
}
if(count==n)
break;
}
execute();
total_time++;
while(burst_time!=0 && count==n)
{
execute();
total_time++;
}
if(count==n)
break;
}
printf("\n\nProcesses In Order To Their Completion (FINAL PROCESS EXECUTION TABLE)\n");
printf(".....\n");
printf("    PROCESS ID    ARRIVAL TIME    SERVICE TIME    WAITING TIME\n");
printf(".....\n");
for(l=0;l<n;l++)
{
```

```
    for(j=0;j<n;j++)
    {
    if(result[l].processID==pcreate[j].processID)
    {
    printf("                P%d                %d                %d
%d\n",result[l].processID,pcreate[j].arrivalTime,pcreate[j].burstTime,result[l].waitTime);
    break;
    }
    }
    averageWaitTime+=(result[l].waitTime);
    }
    printf("AVERAGE  WAITING  TIME :%f\n",averageWaitTime/n);
getch();
}
```

Output Screen Shots:-

Enter No of Processes To Be Created

Enter Arrival and Burst Time of Processes

```
Enter Arrival time of process[1]: 0

Enter Burst time of process[1]: 4

Enter Arrival time of process[2]: 1

Enter Burst time of process[2]: 1

Enter Arrival time of process[3]: 2

Enter Burst time of process[3]: 2

Enter Arrival time of process[4]: 3

Enter Burst time of process[4]: 1_
```

Displaying The Total Waiting Time of All Processes And Average W.T.

```
.
.
Values Entered are : (Table Sorted According To The Arrival Time)
PROCESS TABLE
.........................................
 PROCESS ID          ARRIVAL TIME       SERVICE TIME
.........................................
 P1                0                  4
 P2                1                  1
 P3                2                  2
 P4                3                  1


Processes In Order To Their Completion (FINAL PROCESS EXECUTION TABLE)
.....................................................................

     PROCESS ID     ARRIVAL TIME     SERVICE TIME     WAITING TIME
.....................................................................

        P2              1              1               1
        P1              0              4               1
        P3              2              2               3
        P4              3              1               4
AVERAGE  WAITING  TIME :2.250000
```


THANK
YOU!