Page No.
Date 16 05 24

3) Write a C program to simulate the following preemptive CPU scheduling algorithm to find turnaround time and waiting time

1) FCFS
2) SJF (Non-Preemptive)

```
#include <stdio.h>
void main()
{
 int p[10], at[10], bt[10], ct[10], tat[10], wt[10], i, j,
 temp=0, n;
 float awt=0, atat=0;
 printf("enter the no of process:");
 scanf("%d", &n);
 printf("enter %d process:", n);
 for(i=0; i<n; i++)
 {
   scanf("%d", &p[i]);
 }
 Printf("enter the %d arrival time:", n);
 for(i=0; i<n; i++)
 {
   scanf("%d", &at[i]);
 }
 Printf("enter the %d burst time:", n);
 for(i=0; i<n; i++)
 {
   scanf("%d", &bt[i]);
 }
 ct[0] = at[0] + bt[0];
```

```
for (i=1;i<n;i++)
{
  temp=0;
  if (ct[i-1] < at[i])
  {
    temp = at[i]-ct[i-1];
  }
  ct[i] = ct[i-1] + bt[i] + temp;
}
printf("\n P\t AT\t Bt\t ct\t TAT\t WT");
for(i=0;i<n;i++)
{
tat[i] = ct[i] - at[i];
wt[i] = tat[i] - bt[i];
atat += tat[i];
awt += wt[i];
}
atat = atat/n;
awt = awt/n;
for (i=0;i<n;i++)
{
 printf("\n P %d \t %d \t %d \t %d \t %d \t %d",
        p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
}
printf("\n average turnaround time is %f", atat);
printf("\n average waiting time is %f", awt);
}
```

Page No.
Date

Output

P	A.T	B.T	C.T	TAT	WT
P_1	0	2	2	2	0
P_2	1	2	4	3	1
P_3	5	3	8	3	0
P_4	6	4	12	6	2

average turnaround time is 3.5ms

average waiting time is 0.7ms.

2)

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void swap (int *x, int *y)
{
    int temp = *x;
    *x = *y;
    *y = temp;
}
void sortat (int p[], int at[], int bt[], int n)
{
    int i, j;
    for (i=0; i<n; i++)
    {
        for (j= i+1; j<n; j++)
        {
            if (at[i] > at[j])
            {
                swap (&p[i], &p[j]);
                swap (&at[i], &at[j]);
                swap (&bt[i], &bt[j]);
            }
            else if (at[i] == at[j])
            {
                if (bt[i] > bt[j])
                {
                    swap (&p[i], &p[j]);
                    swap (&at[i], &at[j]);
                    swap (&bt[i], &bt[j]);
                }
            }
        }
    }
}
```

```
void totwt (int btC], int atC], int btC], int totC], int wtC
           int n)
{
 int i;
 for(i=0 ; i<n ; i++)
 {
  tat[i] = bt[i] - at[i];
  wt[i] = tat[i] - btC i];
 }
}
int main()
{
 int *p, *bt, *at, *tat, *wt, *ct, pos, i,j,
  min=1000, n;
 float awt=0; at at=0;
 printf("\n enter the no of processes:");
 scanf("%d", &n);
 p = (int*) malloc (n* sizeof(int));
 at = (int*) malloc (n* sizeof(int));
 bt = (int*) malloc (n* sizeof(int));
 ct = (int*) malloc (n* sizeof(int));
 wt = (int*) malloc (n* sizeof(int));
 tat = (int*) malloc (n* sizeof(int));
 printf("enter the process \n");
 for (i=0; i<n; i++)
 {
  scanf("%d", &p[i]);
 }
Printf(" enter the arrival time \n");
for(i=0; i<n; i++)
{
  scanf("%d", &at[i]);
}
```

```
printf("enter the burst time\n");
for(i=0;i<n;i++);
{
  scanf("%d\n", &bt[i]);
}
sortat(p,at,bt,n);
ct[0] = at[0] + bt[0];
for(i=1;i<n;i++)
{
  for(j=1;j<n;j++)
  {
    if(at[j] <= ct[i-1])
    {
      if(bt[j] < min)
      {
        min = bt[j];
        pos = j;
      }
    }
  }
swap(&p[i], &p[pos]);
swap(&at[i], &at[pos]);
swap(&bt[i], &bt[pos]);
min = 1000;
ct[i] = ct[i-1] + bt[i];
}
for wt(ct, at, bt, tat, wt, n);
Printf("\n P\t at\t bt\t ct\t tat\t wt");
for(i=0;i<n;i++)
{
printf("\n %d\t %d\t %d\t %d\t %d",
      p[i], at[i], bt[i], ct[i], tat[i], wt[i]);
}
```

```
for(i=0;i<n;i++)
{
   atot+=tat[i];
   awt+=wt[i];
}
atat=atat/n;
 awt=awt/n;
 Printf("\n avg tat=%.2f and avg wt=%.2f",
        atat,awt);
  return 0;
}
```

enter the number of process: 4

enter the process

1

2

3

4

Enter the Arrival time

0

1

4

6

enter the burst time

2

6

4

2

P	at	bt	ct	tat	wt
1	0	3	3	3	0
2	1	6	9	8	2
3	6	2	11	5	3
4	4	4	15	11	7

16/5/2024 avg tat = 6.75 ms avg wt = 3.00 ms

Date

//SRTF

```
#include<stdio.h>
void main()
{
  int a[10], b[10], x[10];
  int waiting[10], turnaround[10], completion[10];
  int i, j, smallest, count=0, time, n;
  double avg=0, tt=0, end;
  printf("\n Enter the number of processes");
  scanf("%d", &n);
  for(i=0; i<n; i++)
  {
   printf("\n Enter arrival time of process %d", i+1);
   scanf("%d", &a[i]);
  }
  for(i=0; i<n; i++)
  {
   Printf("\n Enter burst time of process %d:", i+1);
   scanf("%d", &b[i]);
  }
  for(i=0; i<n; i++)
  x[i] = b[i];
  b[9] = 9999;
  for(time=0; count!=n; time++)
  {
    smallest=9;
    for(i=0; i<n; i++)
    {
      if(a[i]<=time && b[i]<b[smallest] && b[i]>0)
      smallest=i;
    }
```

```
b[smallest]--;
if (b[smallest]==0)
{
    count++;
    end = time+1;
    completion[smallest] = end;
    waiting[smallest] = end - a[smallest] - x[smallest];
    turnaround[smallest] = end - a[smallest];
  }
}
printf("\n pid \t burst \t arrival \t waiting \t turnaround \t completion");
for(i=0; i<n; i++)
{
  Printf("\n %d \t %d \t %d \t \t %d \t \t %d \t \t %d", i+1,
        x[i], a[i], waiting[i], turnaround[i],
        completion[i]);
  avg = avg + waiting[i];
  tt = tt + turnaround[i];
}
printf("\n %lf %lf", avg, tt);
printf("\n\n Average waiting time = %lf \n", avg/n);
printf("Average turn around time = %f \n", tt/n);
```

Output: Enter the no of processes 4

Enter arival time of process 1: 0
Enter arival time of process 2: 1
Enter arival time of process 3: 4
Enter arival time of process 4: 6
Enter burst time of process 1: 3
Enter burst time of process 2: 6
Enter burst time of process 3: 4
Enter burst time of process 4: 2

Pid	burst	arrival	waiting	turnaround	comp time
1	3	0	0	3	3
2	6	1	8	14	15
3	4	4	0	4	8
4	2	6	2	4	10

Average waiting time = 2.500
Average Turnaround time = 6.0500

Page No.
Date

Round Robbin Algoritham

```
#include <stdio.h>
#define TQ 3
void main()
{ int rp, ct, rt[10]
  int i,n;
  int bt[10], at[10], ct[10], tat[10], wt[10];
  float atat, awt ;
  atat=0;
  awt=0;
  tat[0]=0;
  wt[0]=0;
  printf("Enter the number of processes \n");
   scanf("%d", &n);
  printf(" Enter the  burst time \n");
  for(i=0; i<n ; i++)
  {
     scanf("%d ", &bt[i]);
   }
  printf("enter the arrival time \n");
  for(i=0; i<n ; i++)
  {
     scanf("%d", &at[i]);
   }
  for(i=0; i<n ; i++)
  {
    rt[i]= bt[i];
  }
  ct = 0;
while( rp >0)
{
```

```
for(i=0; i<n; i++)
{
  if(bt[i]<=tq)
  {
    current_time+= bt[i];
    rt[i]=0;
    rp--;
    Printf("the process %d is completed at %d ms\n",
          i+1, current_time);
  }
  else
  {
    current_time+=tq;
    rt[i]= tq;
    printf("the process %d executed for tq 3ms
          and the remaining time %d\n",
          i+1, rt[i]);
  }
}
printf("process at bt wt tat\n");
for(i=0; i<n; i++)
{
  temp[0]=0;
  temp[i+1] = temp[i] + bt[i];
  wt[i] = temp[i] - at[i];
  tat[i] = wt[i] + bt[i];
  atat += tat[i];
  awt += wt[i];
  Printf("%d %d %d %d %d\n", i+1, at[i], bt[i],
        wt[i], tat[i]);
}
}
```

output:- Enter the Number of Process: 5
Enter time quantam: 2ms

AT of Process1 : 2
BT of process1 : 1
AT of process2 : 1
BT of process2 : 3
AT of process3 : 3
BT of process3 : 2
AT of process4 : 4
BT of process4 : 3
AT of process5 : 0
BT of process5 : 5

ProcessNO	AT	BT	CT	TAT	WT
P1	0	1	14	14	9
P2	1	3	12	11	8
P3	2	1	5	3	2
P4	3	2	7	4	2
P5	4	3	13	9	6

Average Turnaround Time = 8.20000
Average Waiting time = 5.40000

Page No.
Date

Priority (Non-Premptive)

```
#include <stdio.h>
#define MAX 999;
struct proc
{
  int no, at, bt, ct, wt, tat, pri, status;
};
struct proc read(int i)
{
  struct proc p;
  printf("\n Process no: %d \n", i);
  p.no=i;
  printf("enter Arrival time:");
  scanf("%d", &p.at);
  printf("enter Burst time:");
  scanf("%d", &p.bt);
  Printf("Enter Priority:");
  scanf("%d", &p.pri);
  p.status=0;
  return p;

Void main()
{
  int n, s, ct=0, remaining;
  struct proc p[10], temp;
  float avgtat=0, avgwt=0;
  printf("<--Smallest priority First Scheduling Algorithm
        (Non-Premptive)-->\n");
  Printf("enter the number of Processes:");
  scanf("%d", &n);
```

```
for(i=0;i<n;i++)
{
  for(j=0;j<n-i-1;j++)
  {
    if(p[j].at > p[j+1].at)
    {
      temp = p[j];
      p[j] = p[j+1];
      p[j+1] = temp;
      p[9].pri = MAX;
      remaining = n;
      printf("\n Process\t At \t Bt \t Pri \t CT \t TAT
                \t WT \n");
    }
  }
}
for (ct = p[0].at; remaining != 0; )
{
  s=9;
  for(int i=0; i<n; i++)
  {
    if(p[i].at <= ct && p[i].status != 1 && p[i].pri
                                < p[s].pri)
    {
      s=i;
      p[s].ct = ct + p[s].bt;
      p[s].tat = p[s].ct - p[s].at;
      avgtat += p[s].tat;
      p[s].wt = p[s].tat - p[s].bt;
      avgwt += p[s].wt;
      p[s].status = 1;
      remaining--;
```

```
printf("[illegible]",
        [illegible]);
}
avgtat /= n, avgwt /= n;
printf("\n Average Turnaround Time = %f \n Average
        waiting time = %f", avgtat, avgwt);
}
```

output:

Enter no of process: [illegible]
Enter Arrival time: 3
Enter Burst time: 5
Enter Priority: 2

Process no 2
Enter Arrival time: 2
Enter Burst time: 2
Enter priority: 3

Process no 3
Enter Arrival time: 0
Enter Burst time: 3
Enter Priority: 5

Process no 4
Enter Arrival time: 4
Enter Burst time: 4
Enter Priority: 4

Process No: 5

Enter arrival time: 6

Enter Burst time: 1

Enter priority: 1

Process No	AT	BT	Pri	CT	TAT	WT	RT
P_1	0	3	5	3	3	0	0
P_2	0	1	1	9	3	2	2
P_3	3	5	2	8	5	0	0
P_4	2	2	3	11	9	7	7
P_5	4	4	4	15	11	7	7

Average Turn Around Time = 6.200

Average Waiting Time = 3.000

5 Priority (Preemptive)

```
#include<stdio.h>
#define MAX 9999;
Structure proc
{
  Int no, at, bt, rt, ct, wt, tat, pri, temp;
};
Struct proc read(int i)
{
  Struct proc p;
  printf("\n Process No: %d \n", i);
  P.no = i;
  printf(" Enter Arrival Time:");
  Scanf("%d", &P.at);
  printf(" Enter Burst Time:");
  scanf("%d", &P.bt);
  P.rt = p.bt;
  Print f(" Enter priority:");
  Scanf("%d", &P.pri);
  p.temp p = P.pri;
  return p;
}
Void main()
{
  int i,n, remaining, min_val, min_index;
  Struct proc p[10], temp;
  float avgtat = 0, avgwt = 0;
  print f("<-- Smallest Priority First Scheduling
          Algorithm (Preemptive) -->\n");
  printf(" Enter total processes");
  Scanf("%d", &n);
```

```
for(int i=0; i<n; i++)
{
    P[i] = read(i+1);
    remaining = n;
    for(int i=0; i<n-1; i++)
    {
        for(int j=0; j<n-i-1; j++)
        {
            if(P[j].at > P[j+1].at)
            {
                temp = P[j];
                P[j] = P[j+1];
                P[j+1] = temp;
            }
        }
    }
    min_val = P[0].temp, min_index = 0
    for(int j=0; j<n && P[j].at <= P[0].at; j++)
    {
        if(P[j].temp < min_val)
        {
            min_val = P[j].temp, min_index = j;
        }
        i = min_index;
        C = P[i].ct = P[i].at+1;
        P[i].rt--;
        if(P[i].rt == 0)
        {
            P[i].temp = MAX;
            remaining--;
        }
```

```
while (remaining>0)
{
  min.val = p[0].temp, min.index=0;
  for(int j=0; j<n && p[j].at<=c; j++)
  {
    if (p[j].temp < min.val)
    {
      min.val = p[j].temp, min.index=j;
    }
    i = min.index;
    p[i].rt--;
    if(p[i].rt==0)
    {
      p[i].temp=MAX;
      remaining--;
    }
  }
  printf("\n Process\tAT\tBT\tPri\tCT\tTAT\t
        WT\n");
  for(int i=0; i<n; i++)
  {
    p[i].tat = p[i].ct - p[i].at;
    avgtat += p[i].tat;
    p[i].wt = p[i].tat - p[i].bt;
    avgwt += p[i].wt;
    printf("%d\t\t%d\t%d\t%d\t%d\t%d\t%d\n",
        p[i].no, p[i].at, p[i].bt, p[i].pri, p[i].ct,
        p[i].tat, p[i].wt);
  }
  avgtat /= n, avgwt /= n;

  Printf("\n Average Turnaround time = %f\n Average Waiting
        time = %f", avgtat, avgwt);
}
```

output:

Enter number of process: 5
Process no: 1
Enter Arival Time: 0
Enter Burst Time: 3
Enter Priority: 5
Process no: 2
Enter Arival Time: 2
Enter Burst Time: 2
Enter Priority: 3
Process no: 3
Enter Arival Time: 2
Enter Burst Time: 5
Enter Priority: 2
Process no: 4
Enter Arival Time: 4
Enter Burst Time: 4
Enter Priority: 4

Process No	AT	BT	Pri	CT	TAT	WT
P_1	0	3	5	15	15	12
P_2	2	2	3	10	8	6
P_3	2	5	2	8	6	1
P_4	4	4	4	14	10	6
P_5	6	1	1	7	1	0

Average Turnaround time = 8.000000
Average Waiting time = 5.000000