LAB-2

1.) SRTF (SJF Preemptive):-

```
#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("\nEnter 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("pid \t burst \t arival \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 %.1f %.1f", avg, tt);
printf("\n\n Average waiting time = %.2f\n", avg/n);
printf("Average Turnaround time = %.2f", tt/n);
}
```

Output:-

Enter the number of process: 4
Enter arrival time of process 1: 0
Enter arrival time of process 2: 1
Enter arrival time of process 3: 2
Enter arrival time of process 4: 3
Enter burst time of process 1: 8
Enter burst time of process 2: 4
Enter burst time of process 3: 9

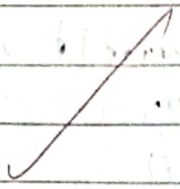
Enter burst time of process : 4 : 5

pid	burst	arrival	Waiting	Turnaround	Completing
1	8	0	9	17	17
2	4	1	0	4	5
3	9	2	15	24	26
4	5	3	2	7	10

If If -

Average waiting time = 6.500000

Average Turnaround time = 13.000000



2) Non-preemptive priority Scheduling :-

```
#include <stdio.h>
#define MAX 9999;
Struct proc
{
    int no, ad, bd, cd, wt, tat, pri, Status;
};
Struct proc read(int i)
{
    Struct proc p;
    printf("\n proattno: %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, a, cd=0, remaining;
    Struct proc p[10], temp;
    float avgtat=0, avgwt=0;
    printf("<-- Smallest Priority First Scheduling
        Algorithm (Non-preemptive) -> \n");
    printf(" Enter number of proattes:");
    Scanf(" %d", &n);
    for(int i=0; i<n; i++)
```

```
        p[i] = read (i+1);
for(int i=0 ; i<n-1 ; i++)
    for(int j=0; j<n-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 No \t AT \t BT \t pri \t CT \t TAT \t WT
        \t RT \n");
for(ct = p[0].at ; remaining != 0)
{
    a = 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 = 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("\n p%d \t %d \t %d \t %d \t %d \t %d \t %d \t %d \t %d",
        p[s].no, p[s].at, p[s].bt, p[s].pri, p[s].ct,
        p[s].tat, p[s].wt, p[s].wt);
}
avg tat/n = %f avgwt/n = %f;
printf("\n Average Turnaround Time = %f \n Average
        Waiting Time = %f", avgtat, avgwt);
}
```

PAGE NO :
DATE :

Output :

```
Smallest priority First Scheduling Algorithm(Non-preemptive)
Enter number of process : 5
process NO: 1
Enter Arrival time: 0
Enter Burst time: 3
Enter priority : 5
process NO: 2
Enter Arrivaltime: 2
Enter Burst time: 2
Enter priority: 3
process NO: 3
Enter Arrival time: 3
Enter Burst time: 5
Enter priority: 2
process NO: 4
Enter Arrival time: 4
Enter Burst time: 4
Enter priority: 4
process NO: 5
Enter Arrivaltime: 6
Enter Burst time: 1
Enter priority : 1
```

Process No	AT	BT	Pri	CT	TAT	WT	RT
P1	0	3	5	3	3	0	0
P3	3	5	2	8	5	0	0
P5	6	1	1	9	3	2	2
P2	2	2	3	11	9	7	7
P4	4	4	4	15	11	7	7

Average Turnaround Time = 6.200000
Average Waiting Time = 3.200000

PAGE NO :
DATE :

3) Priority - preemptive Scheduling :-

```
#include <stdio.h>
# define MAX 9999;
Struct 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;
    printf ("Enter priority:");
    Scanf ("%d", &p.pri);
    p.temp = p.pri;
    return p;
}
Void main ()
{
    int i, n, c, remaining, min_val, min_index;
    Struct proc p[10], temp;
    float avgtat = 0, avgwt = 0;
    printf ("--- Smallest priority First Scheduling
            Algorithm (preemptive) --->\n");
    printf ("Enter Number of process:");
    Scanf ("%d", &n);
```

```
for(int i=0;i<n;i++)
    p[i]=rand()%10;
remaining=n;
for(int i=0;i<n-1;i++)
    for(int j=0;j<n-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].rt=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].ct=c=c+1;
    p[i].rt--;
    if(p[i].rt==0)
```

```
        }
        p[i].temp = MAX;
        remaining --;
    }
}
printf("\nProcNO\tAT\tBT\tPri\tCT\tTAT\tWT");
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("\n P%d\t%d\t%d\t%d\t%d\t%d\t%d",p[i].no, p[i].at, p[i].bt,
           p[i].pri, p[i].ct, p[i].tat, p[i].wt);
}
avg tat /= n, avgwt /= n;
printf("\n Average Turnaround Time = %f\n Average
        Waiting Time = %f", avgtat, avgwt);
}
```

Output:

Enter Number of Processes: 5
ProcNo: 1
Enter Arrival time: 0
Enter Burst time: 3
Enter priority: 5

ProcNo: 2
Enter Arrival time: 2
Enter Burst time: 2
Enter priority: 3

Proall No : 3
Enter Arrival time: 3
Enter burst time: 5
Enter priority: 2

proall No: 4
Enter Arrival time: 4
Enter burst time: 4
Enter priority: 4

proall No: 5
Enter Arrival time: 6
Enter burst time: 1
Enter priority: 1

ProcessNo	AT	BT	Pri	C.T	TAT	WT
P1	0	3	5	15	15	12
P2	2	2	3	10	8	6
P3	3	5	2	9	6	1
P4	4	4	4	14	10	6
P5	6	1	1	7	1	0

Average TurnAround Time = 8.000000
Average waiting Time = 5.000000

3/5/24

4) Round-Robin Scheduling:

```
#include <stdio.h>

Struct proc
{
    int no, at, bt, ct, tat, wt, rt;
};
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;
    return p;
}
int main()
{
    Struct proc p[10], tmp;
    float avgtat=0, avgwt=0;
    int n, tq, ct=0, flag=0, remaining;
    printf("<--- Round Robin Scheduling Algorithm --->
        \n");
    printf("Enter Number of processes: ");
    Scanf("%d", &n);
    printf("Enter time Quantum: ");
    Scanf("%d", &tq);
    for(int i=0; i<n; i++)
        p[i] = read(i+1);
```

```
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)
        {
            tmp=p[j];
            p[j]=p[j+1];
            p[j+1]=tmp;
        }
remaining=n;
printf("\n ProcessNo\tAT\tBT\tCT\tTAT\tWT\n");
for(int i=0; remaining!=0;)
{
        if(p[i].rt<=tq && p[i].rt>0)
        {
            ct+=p[i].rt;
            p[i].rt=0;
            flag=1;
        }
        else if(p[i].rt>0)
        {
            p[i].rt-=tq;
            ct+=tq;
        }
        if(p[i].rt==0 && flag==1)
        {
            flag=0;
            remaining--;
            p[i].ct=ct
            p[i].tat=p[i].ct-p[i].at;
            avg tat+=p[i].tat;
            p[i].wt=p[i].tat-p[i].bt;
            avg wt+=p[i].wt;
```

```
printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t%d\n",
        p[i].no, p[i].at, p[i].bt, p[i].ct,
        p[i].tat, p[i].wt);
}
if(i == n-1 && p[i+1].at <= ct)
    i++;
else
    i=0;
}
avgtat /= n, avgwt /= n;
printf("\nAverage Turnaround Time = %f\nAverage
    Waiting Time = %f", avgtat, avgwt);
}
```

Output:

```
Enter Number of Processes: 4
Enter Time Quantum: 5
process No: 1
Enter Arrival time: 0
Enter Burst Time: 21
process No: 2
Enter Arrival time: 0
Enter Burst time: 3
process No: 3
Enter Arrival time: 0
Enter burst time: 6
process No: 4
Enter Arrival time: 0
Enter Burst time: 2
```

Process No	AT	BT	CT	TAT	WT
P2	0	3	8	8	5
P4	0	2	15	15	13
P3	0	6	21	21	15
P1	0	21	32	32	11

Average TurnaroundTime = 19.000000
Average WaitingTime = 11.000000