30/5/2024

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("\nEnter the number of Processes:");
    scanf("%d", &n);
    for (i=0; i<n; i++)
    {
        printf("\nEnter 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 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 %d", i+1, x[i], a[i], waiting[i], turnaround[i],
        completion[i]);
        avg = avg + waiting[i];
        tt = tt + turnaround[i];
    }
    printf("\n %f %f", avg, tt);
    printf("\n\n Average waiting time = %lf\n", avg/n);
    printf("Average Turnaround time = %lf", tt/n);
}
```

Output :-

Enter the number of Processes : 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

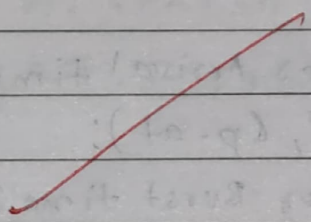
RYA Gold
Page

SURYA Gold
Date Page

```
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	completion
1	8	0	9	17	17
2	4	1	0	4	5
3	9	2	15	24	26
4	5	3	2	7	10

```
Average waiting time : 6.500000
Average Turnaround time : 13.00000
```



x[smallest];
est];

round

H.1.d
od[i],

);
);

```
// Priority scheduling (Non - preemptive)
#include <stdio.h>
# define MAX 9999;
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-preemptive)\n");
    printf("Enter Number of Processes: ");
    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-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("\nProcessNO\tAT\tBT\tPri\tCT\tTAT\
    t WT\tRT\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=ct+p[s].bt;
    p[s].tat=p[s].ct-p[s].at;
    avgtat+=p[s].tat;
    p[s].status=1;
    remaining--;
    printf("p%d\t\t%d\t%d\t%d\t%d\t%d\t%d
        \t%d\t%d\n",p[s].no,p[s].bt,p[s].pri,
        p[s].ct,p[s].tat,p[s].wt,p[s].wt);
}
avgtat/=n, avgwt/=n;
printf("\nAverage TurnAround Time=%f\nAverage
    Waiting Time=%f",avgtat,avgwt);
}
```

Output :-

Smallest Priority first scheduling Algorithm (Non-Preemptive)

Enter Number of Processes : 5

Process NO : 1

Enter Arrival Time : 0

Enter Burst Time : 3

Enter Priority : 5

Process NO : 2

Enter Arrival Time : 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 Arrival Time : 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

SURYA Gold

Date Page

P4 4 4 4 15 11 7 7

Average Turn Around Time = 6.200000

Average Waiting Time = 3.200000

Sun
31/8/24

```
//Priority (Preemptive)
#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("Enter Number of 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;
e = p[i].et = 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].et = c = c+1;
    p[i].rt--;
    if (p[i].rt==0)
    {
        p[i].temp = MAX;
```

```
            remaining--;
        }
    }
    printf("\n ProcessNO \tAT \tBT \tPri \tCT \t
        TAT \tWT\n");
    for(int i=0; i<n; i++)
    {
        p[i].tat = p[i].ct - p[i].at;
        avgtat += p[i].tat;
        avgwt += p[i].wt;
        printf("\n %d \t%d \t%d \t%d \t%d \t%d \t%d
            \n", p[i].no, p[i].at, p[i].bt,
            p[i].bt, p[i].pr, p[i].ct, p[i].tat,
            p[i].wt);
    }
    avgtat /= n, avgwt /= n;
    printf("\n Average TurnAroundTime = %f \n
        Average WaitingTime = %f", avgtat, avgwt);
}
```

Output :-

Enter Nomber of Processes : 5

Process NO : 1

Enter Arrival Time : 0

Enter Burst Time : 3

Enter Priority : 5

Process NO : 2

Enter Arrival Time : 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 : 6
Enter Burst Time : 1
Enter Priority : 1

ProcessNO	AT	BT	Pri	CT	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 waitingTime : 5.000000

```
//Round Robin

#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("Enter Number of Processess:");
    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-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("\nProcessNO \tAT\tBT\tRT\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;
        avgtat+=p[i].tat;
        p[i].wt=p[i].tat-p[i].bt;
        avgwt+=p[i].wt;
```

```
printf("P-%d\t\t%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("Average TurnAround Time = %f\n Average
Waiting Time = %f", avgtat, avgwt);
}
```

Output :-

Enter Number of Proccesses : 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 Turn Around Time = 19.000000
Average Waiting Time = 11.000000

Sw
31/5/24