19BCE0829
KANDRA KSHEERAJ

Scheduling Algorithms

```
kandraksheeraj@srikithadesk-VirtualBox: ~
kandraksheeraj@srikithadesk-VirtualBox:~$ gedit scheduling_algo.c
kandraksheeraj@srikithadesk-VirtualBox:~$ gcc scheduling_algo.c
kandraksheeraj@srikithadesk-VirtualBox:~$ ./a.out


***MENU*
1. FCFS
2. SJF
3. SRTF
4. Priority Scheduling
5. Round Robin
6. Exit
Enter your choice : 1
FCFS Algorithm
**INPUT**
Enter number of process
5
Enter arrival time for processess
4
6
0
6
5
Enter burst time for processess
5
4
3
2
4
```

```
kandraksheeraj@srikithadesk-VirtualBox: ~
**OUTPUT**
Sequence of execution is
[P2]  [P0]  [P4]  [P1]  [P3]
Process  Arrival-time(s)  Burst-time(s)  Waiting-time(s)  Turnaround-time(s)
P0            4              5              0               5
P1            6              4              7               11
P2            0              3              0               3
P3            6              2              11               13
P4            5              4              4               8
Average waiting time(s) 4.400000
Average turnaroundtime(s) 8.000000
CPU idle time(s)1


***MENU*
1. FCFS
2. SJF
3. SRTF
4. Priority Scheduling
5. Round Robin
6. Exit
Enter your choice : 2
SJF (Shortest Job First) - Non Preemptive
Enter Number of Processes
4
Enter Arrival Time & Burst Time for Process P0
0
6
Enter Arrival Time & Burst Time for Process P1
```

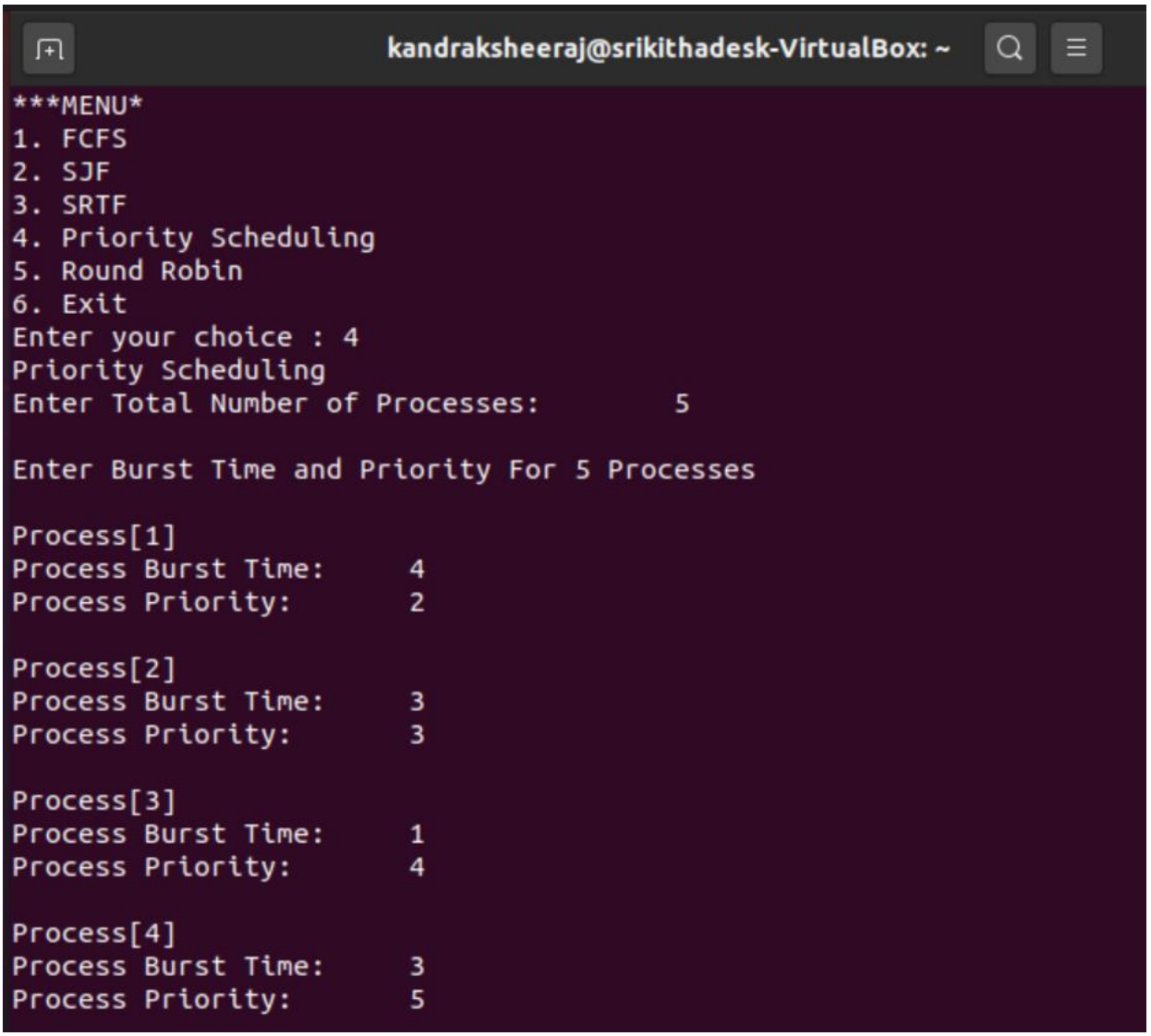
19BCE0829
KANDRA KSHEERAJ

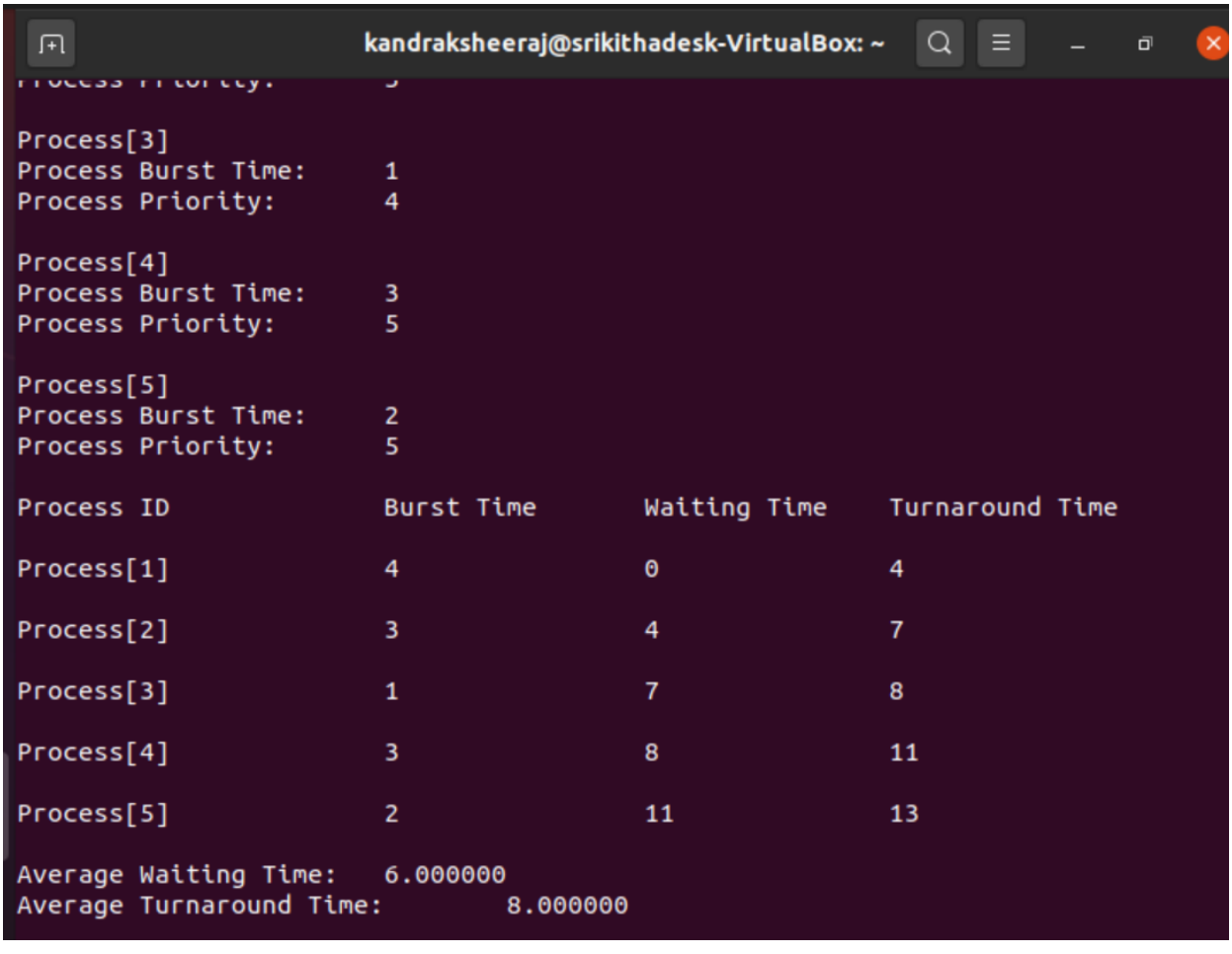
```
kandraksheeraj@srikithadesk-VirtualBox: ~
Enter Arrival Time & Burst Time for Process P1
1
8
Enter Arrival Time & Burst Time for Process P2
2
7
Enter Arrival Time & Burst Time for Process P3
3
3
*************
Pro     ArTi    BuTi    TaTi    WtTi
*************
0       0       6       6       0
1       1       8       9       15
2       2       7       9       7
3       3       3       6       3
*************
Gantt Chart
0 -> [P0] <- 6 -> [P3] <- 9 -> [P2] <- 16 -> [P1] <- 24
*************
Average Waiting Time : 6.25
Average Turnaround Time : 7.50
```

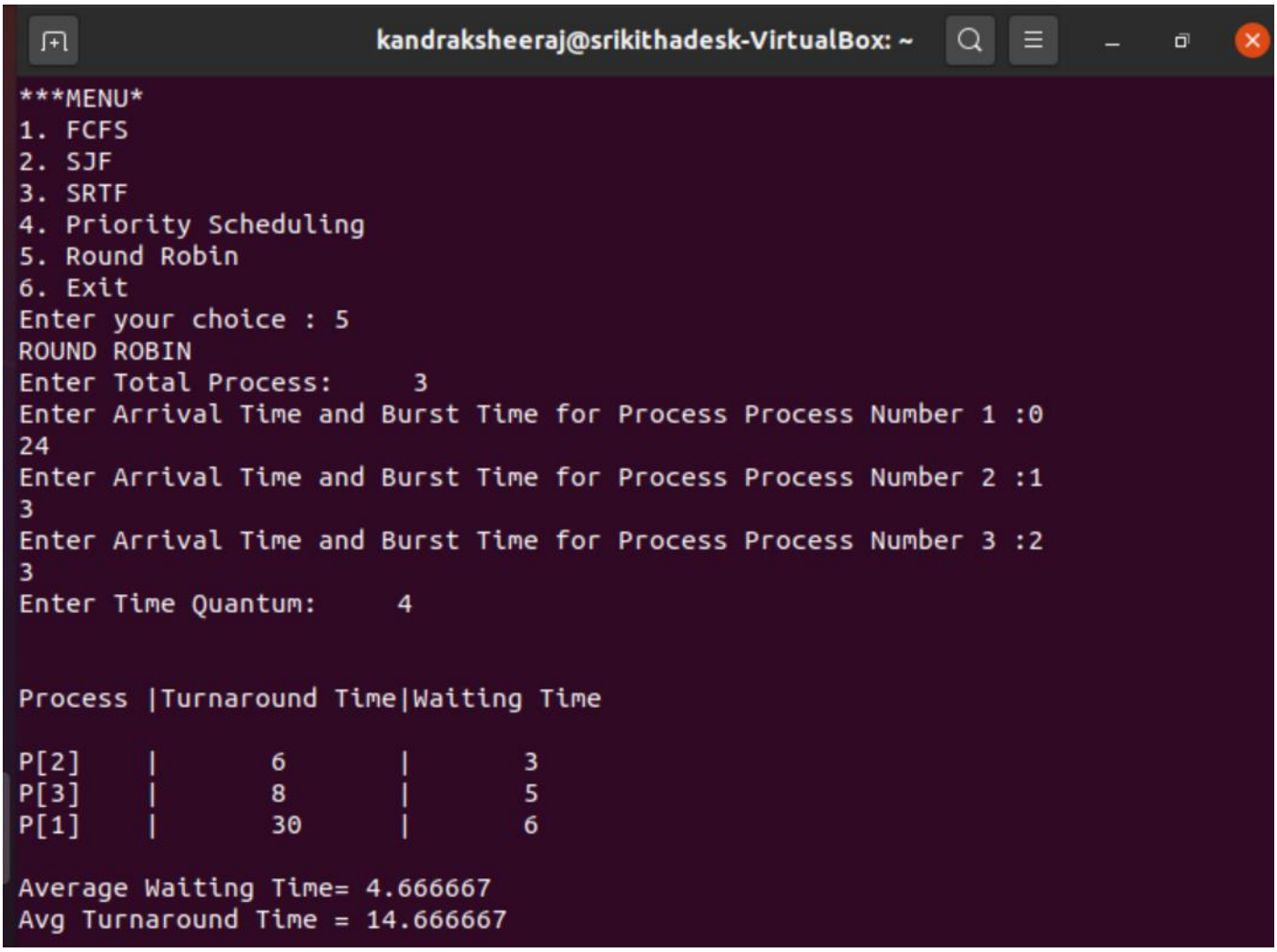
```
kandraksheeraj@srikithadesk-VirtualBox: ~
***MENU*
1. FCFS
2. SJF
3. SRTF
4. Priority Scheduling
5. Round Robin
6. Exit
Enter your choice : 3
SJF (Shortest Job First) - Preemptive
Enter the number of Processes:
4
Enter arrival time
0
1
2
3
Enter burst time
8
4
9
5


Average waiting time = 6.500000
Average Turnaround time = 13.000000
```

19BCE0829
KANDRA KSHEERAJ







Code:

```
#include<stdio.h>

#include<stdlib.h>

#define MAX 100

typedef struct

{

      int pid;

      int arrival_time;

      int burst_time;

      int waiting_time;

      int turnaround_time;

      int completion_time;

      }Process;
```

```
void FCFS()

{

    int
arrival_time[10],arrival_time2[10],burst_time[100],ex[100],seq[100],re[100],wai
ting_time[100],turnaround_time[100];

    int n,i,j,start,pos,max=0,min,idle=0,k=0;

    float av1=0,av2=0;

     printf("FCFS Algorithm\n");

    printf("*****INPUT*****\n");

    printf("Enter number of process\n");

    scanf("%d",&n);

    printf("Enter arrival time for processess\n");

    for(i=0;i<n;i++)

    {

     scanf("%d",&arrival_time[i]);

     arrival_time2[i]=arrival_time[i];

    }

    printf("Enter burst time for processess\n");

    for(i=0;i<n;i++)

    {

     scanf("%d",&burst_time[i]);

    }

    start=arrival_time[0];

    for(i=1;i<n;i++)

    {

      if(start>arrival_time[i])

       {

       start=arrival_time[i];
```

```
    }
  }
 printf("*****OUTPUT*****\n");
 printf("Sequence of execution is\n");
 for(i=0;i<n;i++)
 {
 if(max<arrival_time[i])
  {
   max=arrival_time[i];
  }
 }
 max=max+1;
for(i=0;i<n;i++,k++)
  {  min=max;
    for(j=0;j<n;j++){
        if(arrival_time[j]!=-1)
          {
            if(arrival_time[j]<min)
              {
               min=arrival_time[j];
               pos=j;
              }
           }
      }
   printf("[P%d]  ",pos);
   seq[k]=pos;
   if(start<arrival_time[pos]){
      re[pos]=start;
```

```
        idle+=arrival_time[pos]-start;
        start=arrival_time[pos];
        start+=burst_time[pos];
        arrival_time[pos]=-1;
        ex[pos]=start;
      }
      else{
        re[pos]=start;
        start+=burst_time[pos];
        arrival_time[pos]=-1;
        ex[pos]=start;
       }
     }
    printf("\n");
    for(i=0;i<n;i++)
    {
       turnaround_time[i]=ex[i]-arrival_time2[i];
       waiting_time[i]=turnaround_time[i]-burst_time[i];
    }
 printf("Process  Arrival-time(s)  Burst-time(s)  Waiting-time(s)  Turnaround-time(s)\n");
   for(i=0;i<n;i++)
    {
      printf("P%d            %d              %d             %d              %d\n",i,arrival_time2[i],burst_time[i],waiting_time[i],turnaround_time[i]);
    }
   for(i=0;i<n;i++)
   {
    av1+=turnaround_time[i];
```

```
    av2+=waiting_time[i];
   }
  printf("Average waiting time(s) %f\nAverage turnaroundtime(s) %f\nCPU idle time(s)%d\n",av2/n,av1/n,idle);
}

void SJF()
{
struct time
{
int p,arrival_time,burst_time,waiting_time,turnaround_time,st;
};
int process(struct time a[], int pro,int t)
{
int i,minpro, mintime=999;
for(i=0;i<pro;i++){
if(a[i].arrival_time <= t && a[i].st == 0)
{
if(mintime > a[i].burst_time)
{
mintime = a[i].burst_time;
minpro = i;
}
}
}
a[minpro].st = 1;
return minpro;
}
```

```
void ganttchart(struct time a[],int gc[],int pro)
{
int i,x=0;
printf("Gantt Chart\n");
printf("0");
for(i=0;i<pro;i++)
{
x = x + a[gc[i]].burst_time;
printf(" -> [P%d] <- %d",a[gc[i]].p,x);
}
printf("\n");
return;
}
{
int i,pro,curpro,t=0,gc[100];
struct time a[100];
float avgwt=0,avgtt=0;
printf("SJF (Shortest Job First) - Non Preemptive\n");

printf("Enter Number of Processes\n");
scanf("%d",&pro);
for(i=0;i<pro;i++)
{
printf("Enter Arrival Time & Burst Time for Process P%d\n",i);
a[i].p = i;
scanf("%d%d",&a[i].arrival_time,&a[i].burst_time);
a[i].st = 0;
}
```

```
for(i=0;i<pro;i++)
{
curpro = process(a,pro,t);
a[curpro].waiting_time = t - a[curpro].arrival_time;
a[curpro].turnaround_time = a[curpro].arrival_time + a[curpro].burst_time;
t = t + a[curpro].burst_time;
avgwt = avgwt + a[curpro].waiting_time;
avgtt = avgtt + a[curpro].turnaround_time;
gc[i] = curpro;
}
printf("****************************************\n");
printf("Pro\tArTi\tBuTi\tTaTi\tWtTi\n");
printf("****************************************\n");
for(i=0;i<pro;i++)
{
printf("%d\t%d\t%d\t%d\t%d\n",a[i].p,a[i].arrival_time,a[i].burst_time,a[i].tur
naround_time,a[i].waiting_time);
}
printf("****************************************\n");
ganttchart(a,gc,pro);
printf("****************************************\n");
avgwt = avgwt/pro;
avgtt = avgtt/pro;
printf("Average Waiting Time : %.2f\n",avgwt);
printf("Average Turnaround Time : %.2f\n",avgtt);
}
}
void SRTF()
```

```
{
 printf("SJF (Shortest Job First) - Preemptive\n");
 int a[10],b[10],x[10],i,j,smallest,count=0,time,n;
 double avg=0,tt=0,end;
  printf("Enter the number of Processes:\n");
  scanf("%d",&n);
 printf("Enter arrival time\n");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
 printf("Enter burst time\n");
 for(i=0;i<n;i++)
 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;
   avg=avg+end-a[smallest]-x[smallest];
   tt= tt+end-a[smallest];
  }
 }
 printf("\n\nAverage waiting time = %lf\n",avg/n);
    printf("Average Turnaround time = %lf",tt/n);
    return 0;
}

void Priority()
{
     printf("Priority Scheduling\n");
int burst_time[20], process[20], waiting_time[20], turnaround_time[20],
priority[20];
int i, j, limit, sum = 0, position, temp;
float average_wait_time, average_turnaround_time;
printf("Enter Total Number of Processes:\t");
scanf("%d", &limit);
printf("\nEnter Burst Time and Priority For %d Processes\n", limit);
for(i = 0; i < limit; i++)
{
printf("\nProcess[%d]\n", i + 1);
printf("Process Burst Time:\t");
scanf("%d", &burst_time[i]);
printf("Process Priority:\t");
scanf("%d", &priority[i]);
process[i] = i + 1;
```

```
}
for(i = 0; i < limit; i++)
{
position = i;
for(j = i + 1; j < limit; j++){
if(priority[j] < priority[position])
{
position = j;
}
}
temp = priority[i];
priority[i] = priority[position];
priority[position] = temp;
temp = burst_time[i];
burst_time[i] = burst_time[position];
burst_time[position] = temp;
temp = process[i];
process[i] = process[position];
process[position] = temp;
}
waiting_time[0] = 0;
for(i = 1; i < limit; i++)
{
waiting_time[i] = 0;
for(j = 0; j < i; j++)
{
waiting_time[i] = waiting_time[i] + burst_time[j];
}
```

```
sum = sum + waiting_time[i];

}

average_wait_time = sum / limit;

sum = 0;

printf("\nProcess ID\t\tBurst Time\t Waiting Time\t Turnaround Time\n");

for(i = 0; i < limit; i++)

{

turnaround_time[i] = burst_time[i] + waiting_time[i];

sum = sum + turnaround_time[i];

printf("\nProcess[%d]\t\t%d\t\t %d\t\t %d\n", process[i], burst_time[i],
waiting_time[i],

turnaround_time[i]);

}

average_turnaround_time = sum / limit;

printf("\nAverage Waiting Time:\t%f", average_wait_time);

printf("\nAverage Turnaround Time:\t%f\n", average_turnaround_time);

return 0;

}


void RR()

{

      printf("ROUND ROBIN\n");

  int count,j,n,time,remain,flag=0,time_quantum;

  int wait_time=0,turnaround_time=0,arrival_time[10],burst_time[10],rt[10];

  printf("Enter Total Process:\t ");

  scanf("%d",&n);

  remain=n;

  for(count=0;count<n;count++)

  {
```

```
    printf("Enter Arrival Time and Burst Time for Process Process Number %d
:",count+1);
    scanf("%d",&arrival_time[count]);
    scanf("%d",&burst_time[count]);
    rt[count]=burst_time[count];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];
      rt[count]=0;
      flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;
      time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-arrival_time[count],time-
arrival_time[count]-burst_time[count]);
      wait_time+=time-arrival_time[count]-burst_time[count];
      turnaround_time+=time-arrival_time[count];
```

```
      flag=0;
    }
    if(count==n-1)
      count=0;
    else if(arrival_time[count+1]<=time)
      count++;
    else
      count=0;
  }
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
  printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

  return 0;
}

int main()
{
int ch;
while(1){
printf("\n\n****MENU****\n");
printf("1. FCFS\n");
printf("2. SJF\n");
printf("3. SRTF\n");
printf("4. Priority Scheduling\n");
printf("5. Round Robin\n");
printf("6. Exit\n");
printf("Enter your choice : ");
```

```
scanf("%d",&ch);
switch(ch)
{
case 1:
      FCFS();
      break;
case 2:
      SJF();
      break;
case 3:
      SRTF();
      break;
case 4:
      Priority();
      break;
case 5:
      RR();
      break;
case 6:
      exit(0);
default:
      printf("Invalid input!");
      return 0;
      }
  }
}
```