Operating System

Assessment-2

QUESTION:

Consider a corporate hospital where we have n number of patients waiting for consultation. The amount of time required to serve a patient may vary, say 10 to 30 minutes. If a patient arrives with an emergency, he /she should be attended immediately before others, which may increase the waiting time of other patients. If you are given this problem with the following algorithms how would you devise an effective scheduling so that it optimizes the overall performance such as minimizing the waiting time of all patients. [Single queue or multi-level queue can be used].
• Consider the availability of single and multiple doctors
• Assign top priority for patients with emergency case, women, children, elders, and youngsters.
• Patients coming for review may take less time than others. This can be taken into account while using SJF.
a. FCFS
b. SJF (primitive and non-pre-emptive)
c. Priority
d. Round robin

SOLUTION:

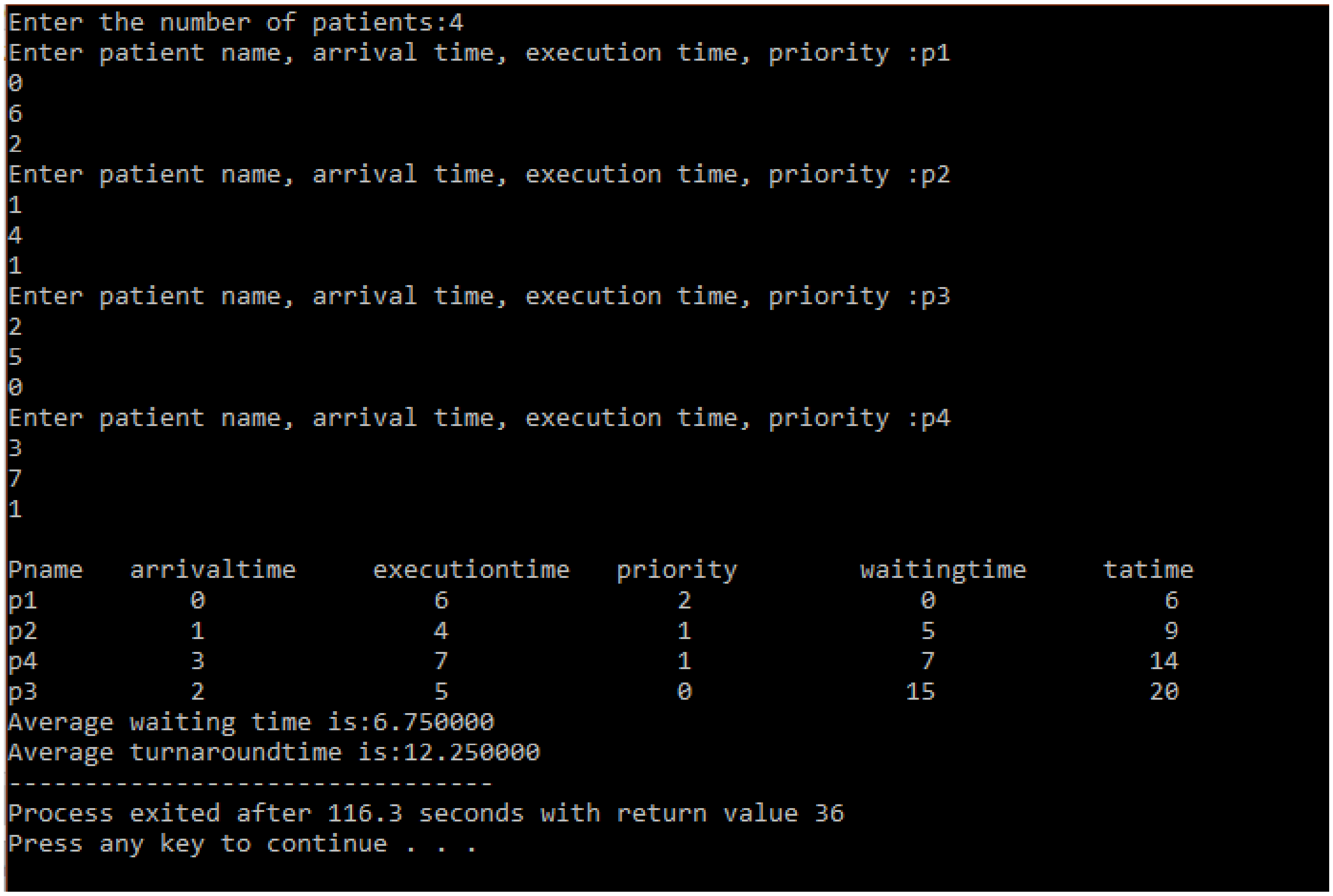
For the given problem I'd choose priority preemptive scheduling algorithm. If

there is
no emergency the patients will be having same priority and would be executed by FCFS algorithm and if there's any emergency the patient will be given more priority and will be executed first i.e., for normal patient **priority=0** and in emergency **priority will increase according to the emergency.**

CODE:

```
#include<stdio.h>
#include<string.h>
void main()
{
   int et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];
   char ch;
   int totwt=0,totta=0;
   float awt,ata;
   char pn[10][10],t[10];
   printf("Enter the number of patients:");
   scanf("%d",&n);
   for(i=0; i<n; i++)
   {

      printf("Enter patient name, arrival time, execution time, priority :"); scanf("%s%d
      %d%d",pn[i],&at[i],&et[i],&p[i]);
   }



   for(i=0; i<n; i++)
      for(j=0; j<n; j+
      +)
      {
         if(p[i]>p[j])
         {
            temp=p[i];
            p[i]=p[j];
            p[j]=temp;
            temp=at[i];
            at[i]=at[j];
            at[j]=temp;
            temp=et[i];
            et[i]=et[j];
            et[j]=temp;
            strcpy(t,pn[i]);
            strcpy(pn[i],pn[j]);
            strcpy(pn[j],t);
         }
      }
   for(i=0; i<n; i++)

   {

      if(i==0)
      {
         st[i]=at[i];
         wt[i]=st[i]-at[i];
         ft[i]=st[i]+et[i];
         ta[i]=ft[i]-at[i];
      }
      else
      {
         st[i]=ft[i-1];
         wt[i]=st[i]-at[i];
         ft[i]=st[i]+et[i];
         ta[i]=ft[i]-at[i];

      }
      totwt+=wt[i];
      totta+=ta[i];
   }

   awt=(float)totwt/n;
   ata=(float)totta/n;
   printf("\nPname\tarrivaltime\texecutiontime\tpriority\twaitingtime\ttatime");
   for(i=0; i<n; i++)
      printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],p[i],wt[i],ta[i]);
   printf("\nAverage waiting time is:%f",awt);
   printf("\nAverage turnaroundtime is:%f",ata);
}
```

OUTPUT:



```
Enter the number of patients:4
Enter patient name, arrival time, execution time, priority :p1
0
6
2
Enter patient name, arrival time, execution time, priority :p2
1
4
1
Enter patient name, arrival time, execution time, priority :p3
2
5
0
Enter patient name, arrival time, execution time, priority :p4
3
7
1

Pname   arrivaltime     executiontime   priority        waitingtime     tatime
p1          0               6               2               0               6
p2          1               4               1               5               9
p4          3               7               1               7              14
p3          2               5               0              15              20
Average waiting time is:6.750000
Average turnaroundtime is:12.250000
--------------------------------
Process exited after 116.3 seconds with return value 36
Press any key to continue . . .
```

a) Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithm for each queue, the criteria used to move processes between queues, and so on. These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets? Implement the below mentioned algorithms and determine the efficiency of each algorithm. (Assume your own data for input) 1. Priority and SJF 2. Multilevel feedback queues and FCFS 3. Priority and FCFS 4. RR and SJF

SOLUTION:

1. Priority and SJF

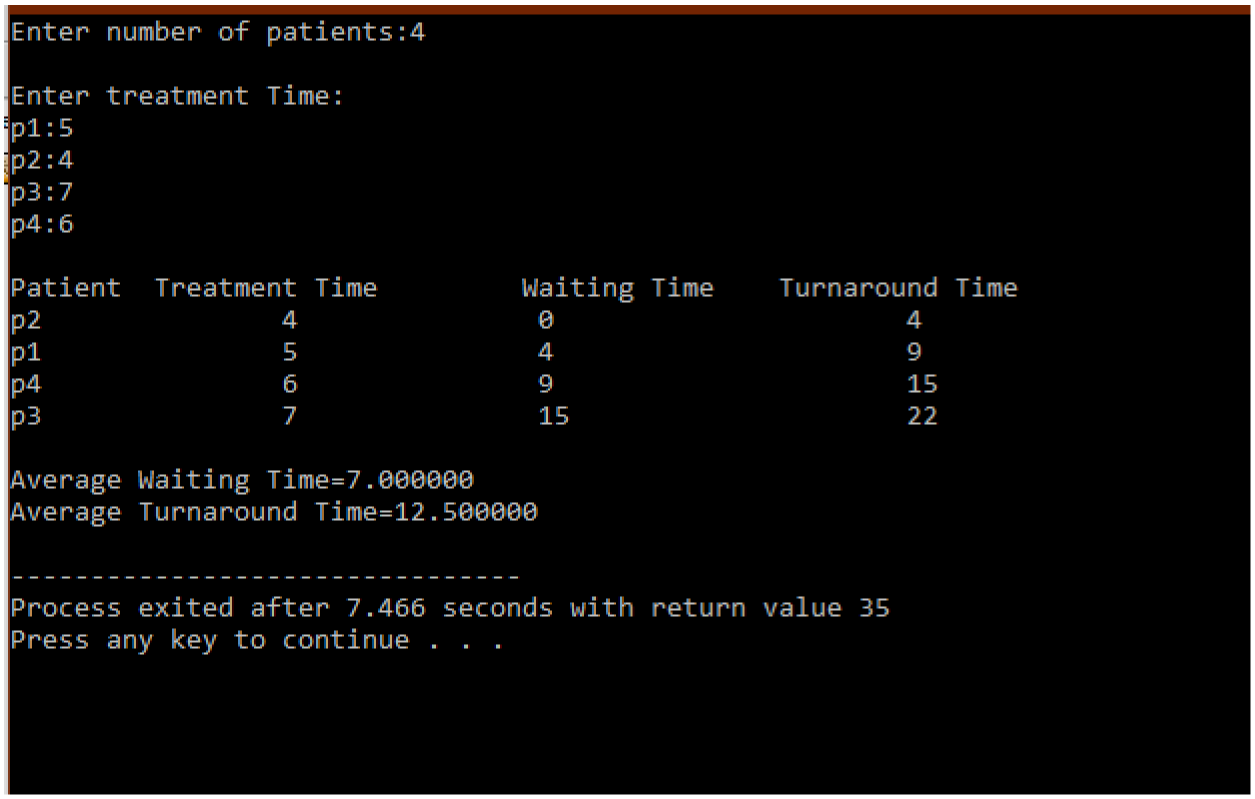
The shortest job has the highest priority

SJF:

```
#include<stdio.h>

void main()
{
   int  bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of patients:");
   scanf("%d",&n);

   printf("\nEnter treatment Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;         //contains process number
   }

   //sorting burst time in ascending order using selection sort
```

```
for(i=0;i<n;i++)
{
   pos=i; for(j=i+1;j<n;j+
   +)
   {
      if(bt[j]<bt[pos])
         pos=j;
   }

   temp=bt[i];
   bt[i]=bt[pos];
   bt[pos]=temp;

   temp=p[i];
   p[i]=p[pos];
   p[pos]=temp;
}

wt[0]=0;         //waiting time for first process will be zero

//calculate waiting
time for(i=1;i<n;i++)
{
   wt[i]=0;
   for(j=0;j<i;j++)
      wt[i]+=bt[j];

   total+=wt[i];
}

avg_wt=(float)total/n;    //average waiting time
total=0;

printf("\nPatient\t    Treatment Time  \tWaiting Time\tTurnaround
Time"); for(i=0;i<n;i++)
{
   tat[i]=bt[i]+wt[i];     //calculate turnaround time
   total+=tat[i];
   printf("\np%d\t\t %d\t\t  %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
```

```
        }

        avg_tat=(float)total/n;    //average turnaround time
        printf("\n\nAverage Waiting Time=%f",avg_wt);
        printf("\nAverage Turnaround Time=%f\n",avg_tat);
    }
```

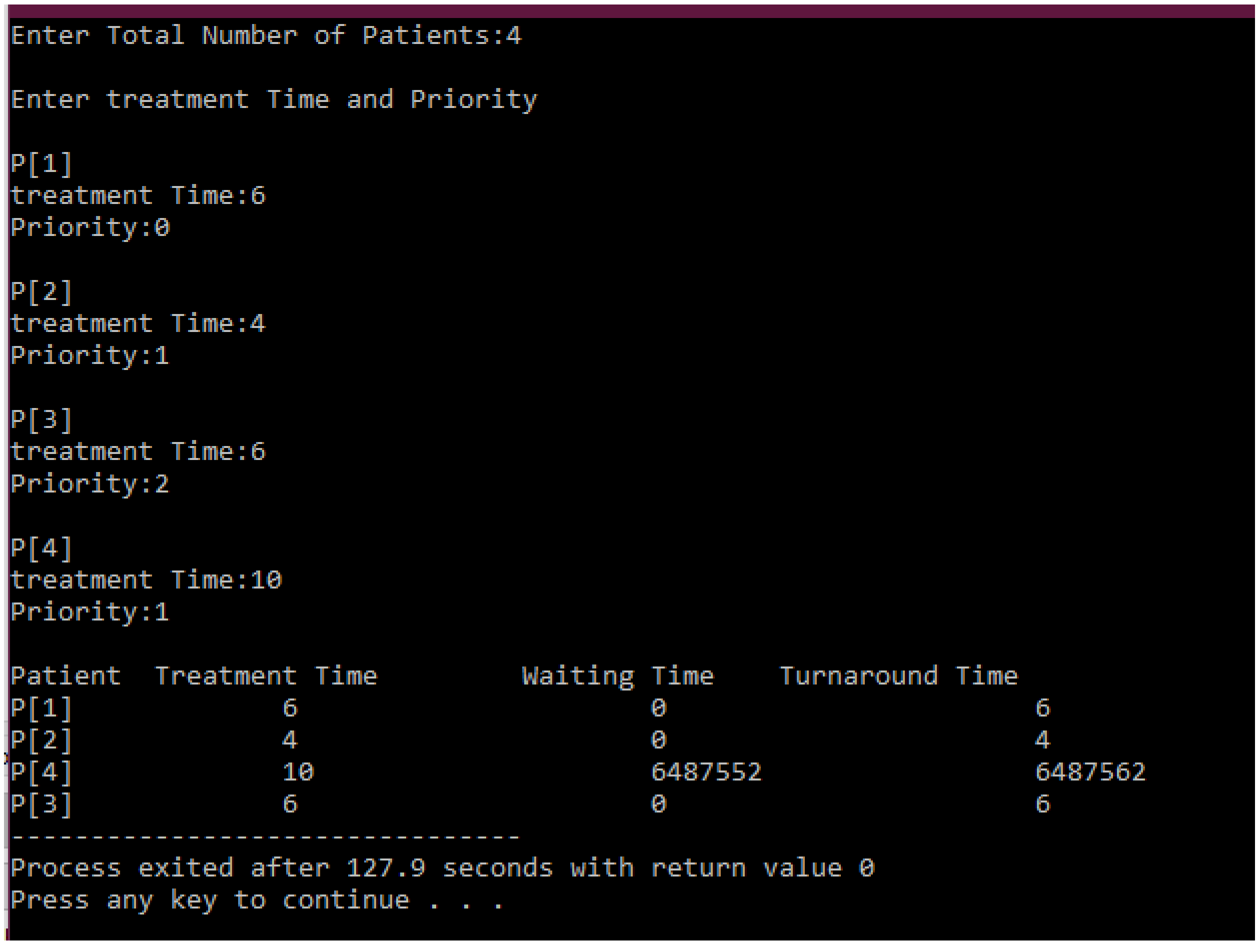


```
Enter number of patients:4

Enter treatment Time:
p1:5
p2:4
p3:7
p4:6

Patient  Treatment Time         Waiting Time    Turnaround Time
p2               4               0                       4
p1               5               4                       9
p4               6               9                       15
p3               7               15                      22

Average Waiting Time=7.000000
Average Turnaround Time=12.500000

--------------------------------
Process exited after 7.466 seconds with return value 35
Press any key to continue . . .
```

PRIORITY:

```
    #include<stdio.h>

    int main()
    {
        int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
        printf("Enter Total Number of Patients:");
        scanf("%d",&n);

        printf("\nEnter treatment Time and Priority\n");
        for(i=0;i<n;i++)
        {
            printf("\nP[%d]\n",i+1);
            printf("treatmentTime"
          );

scanf("%d",&bt[i]);
printf("Priority:");
scanf("%d",&pr[i]);
          p[i]=i+1;       //contains patient number
       }

       //sorting treatment time, priority and patient number in ascending order using
    selection sort
       for(i=0;i<n;i++)
       {
          pos=i; for(j=i+1;j<n;j+
          +)
          {
             if(pr[j]<pr[pos])
                pos=j;
          }

          temp=pr[i];
          pr[i]=pr[pos];
          pr[pos]=temp;

          temp=bt[i];
          bt[i]=bt[pos];
          bt[pos]=temp;

          temp=p[i];
          p[i]=p[pos];
          p[pos]=temp;
       }

       wt[0]=0;   //waiting time for first process is zero

       //calculate waiting
       time for(i=1;i<n;i++)
       {
          wt[i]=0;
          for(j=0;j<i;j++)
             wt[i]+=bt[j];
```

```
        total+=wt[i];
        }
    avg_wt=total/n;    //average waiting time
    total=0;

    printf("\nPatient\t Treatment Time     \tWaiting Time\tTurnaround
    Time"); for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];     //calculate turnaround time
        total+=tat[i];
        printf("\nP[%d]\t\t %d\t\t   %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=total/n;   //average turnaround time
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

    return 0;
}
```



```
Enter Total Number of Patients:4

Enter treatment Time and Priority

P[1]
treatment Time:6
Priority:0

P[2]
treatment Time:4
Priority:1

P[3]
treatment Time:6
Priority:2

P[4]
treatment Time:10
Priority:1

Patient  Treatment Time         Waiting Time    Turnaround Time
P[1]            6                       0                       6
P[2]            4                       0                       4
P[4]            10                      6487552                 6487562
P[3]            6                       0                       6
--------------------------------
Process exited after 127.9 seconds with return value 0
Press any key to continue . . .
```

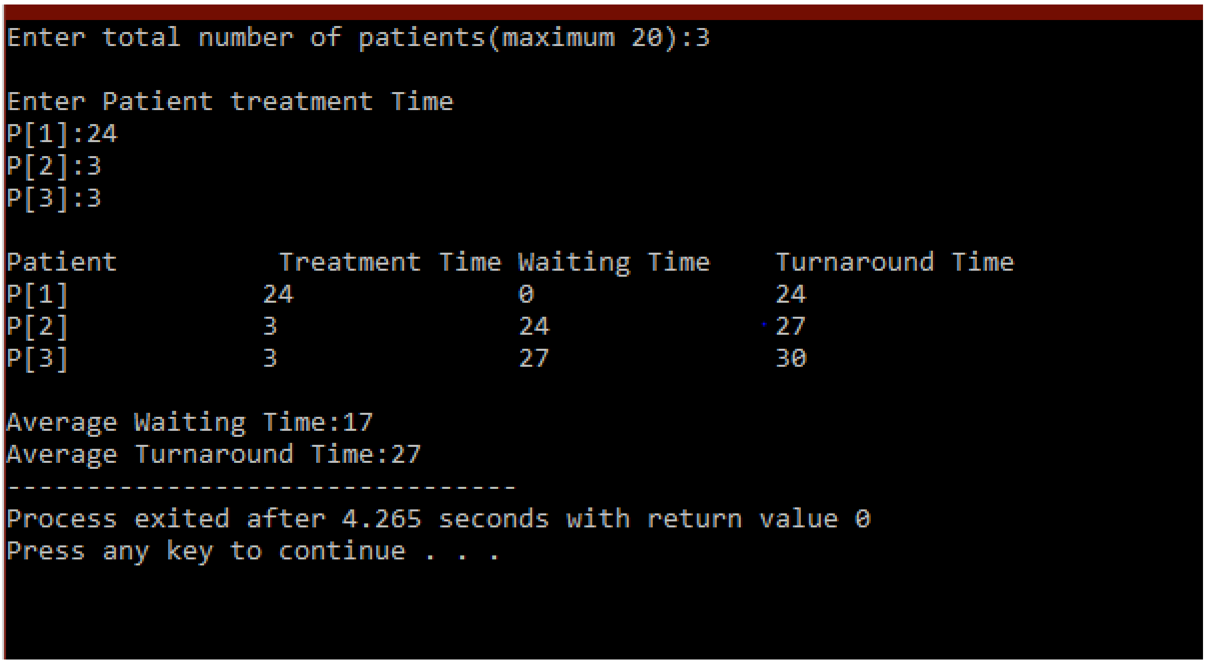
2. Multilevel feedback queues and FCFS

The lowest level of MLFQ is FCFS.

FCFC:

```
#include<stdio.h>

int main()
{
   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
   printf("Enter total number of patients(maximum 20):");
   scanf("%d",&n);

   printf("\nEnter Patient treatment
   Time\n"); for(i=0;i<n;i++)
   {
      printf("P[%d]:",i+1);
      scanf("%d",&bt[i]);
   }
```

```
    wt[0]=0;   //waiting time for first patient is 0

    //calculating waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }

    printf("\nPatient\t\t Treatment Time\tWaiting Time\tTurnaround Time");

    //calculating turnaround time
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
    }

    avwt/=i;
    avtat/=i;
    printf("\n\nAverage Waiting Time:%d",avwt);
    printf("\nAverage Turnaround Time:%d",avtat);

    return 0;
}
```



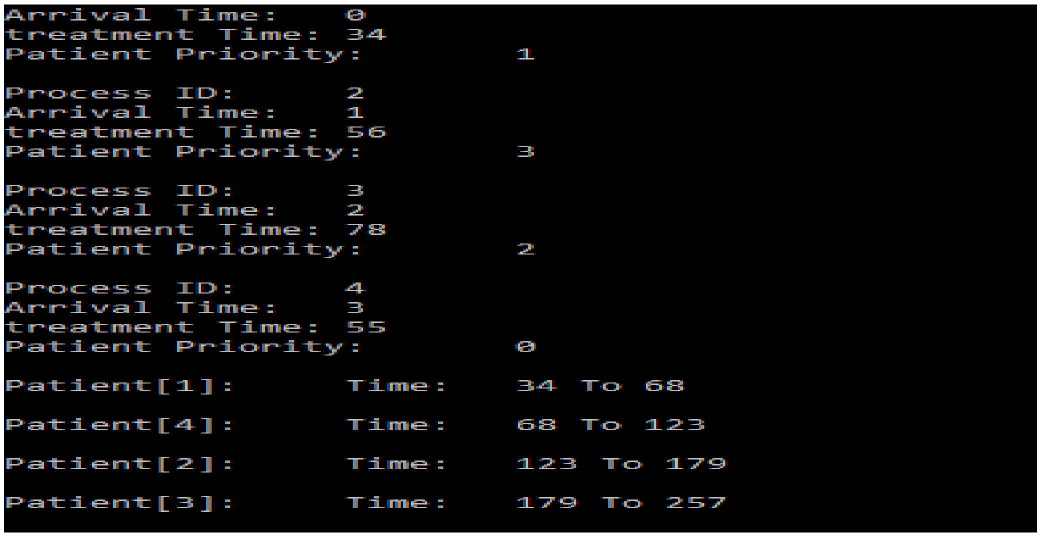
```
Enter total number of patients(maximum 20):3

Enter Patient treatment Time
P[1]:24
P[2]:3
P[3]:3

Patient          Treatment Time Waiting Time    Turnaround Time
P[1]            24              0               24
P[2]            3               24              27
P[3]            3               27              30

Average Waiting Time:17
Average Turnaround Time:27
--------------------------------
Process exited after 4.265 seconds with return value 0
Press any key to continue . . .
```

MLFQ:

```
#include<stdio.h>

#define N 10

typedef struct
{
    int patient_id, arrival_time, treatment_time,
    priority; int q, ready;
}process_structure;

int Queue(int t1)
{
    if(t1 == 0 || t1 == 1 || t1 == 2 || t1 == 3)
    {
        return 1;
    }
    else
    {
        return 2;
    }
}

int main()
{
    int limit, count, temp_process, time, j,
    y; process_structure temp;
    printf("Enter Total Number of Patients:\t");
    scanf("%d", &limit);
    process_structure process[limit];
    for(count = 0; count < limit;
    count++)
    {
        printf("\nProcess ID:\t");
        scanf("%d", &process[count].patient_id);
        printf("Arrival Time:\t");
        scanf("%d",
        &process[count].arrival_time);
        printf("treatment Time:\t");
        scanf("%d",
        &process[count].treatment_time);
        printf("Patient Priority:\t");
        scanf("%d", &process[count].priority);
        temp_process = process[count].priority;
        process[count].q = Queue(temp_process);
        process[count].ready = 0;
    }
    time =
    process[0].treatment_time;
    for(y = 0; y < limit; y++)
    {
        for(count = y; count < limit; count++)
        {
            if(process[count].arrival_time < time)
```

```
{
    process[count].ready = 1;
    }
}
for(count = y; count < limit - 1; count++)
{
    for(j = count + 1; j < limit; j++)
    {
        if(process[count].ready == 1 && process[j].ready == 1)
        {
            if(process[count].q == 2 && process[j].q == 1)
            {
                temp = process[count];
                process[count] =
                process[j]; process[j] =
                temp;
            }
        }
    }
}
for(count = y; count < limit - 1; count++)
{
    for(j = count + 1; j < limit; j++)
    {
        if(process[count].ready == 1 && process[j].ready == 1)
        {
            if(process[count].q == 1 && process[j].q == 1)
            {
                if(process[count].treatment_time > process[j].treatment_time)
                {
                    temp = process[count];
                    process[count] =
                    process[j]; process[j] =
                    temp;
                }
                else
                {
                    break;
                }
            }
        }
    }
}
```

```
            printf("\nPatient[%d]:\tTime:\t%d To %d\n", process[y].patient_id, time, time + 
    process[y].treatment_time);
            time = time + 
            process[y].treatment_time;
            for(count = y; count < limit; 
            count++)
            {
                if(process[count].ready  == 1)
                {
                    process[count].ready  = 0;
                }
            }
        }
        return 0;
    }
```



```
Arrival Time:     0
treatment Time:  34
Patient Priority:         1

Process ID:       2
Arrival Time:     1
treatment Time:  56
Patient Priority:         3

Process ID:       3
Arrival Time:     2
treatment Time:  78
Patient Priority:         2

Process ID:       4
Arrival Time:     3
treatment Time:  55
Patient Priority:         0

Patient[1]:       Time:    34 To 68

Patient[4]:       Time:    68 To 123

Patient[2]:       Time:    123 To 179

Patient[3]:       Time:    179 To 257
```

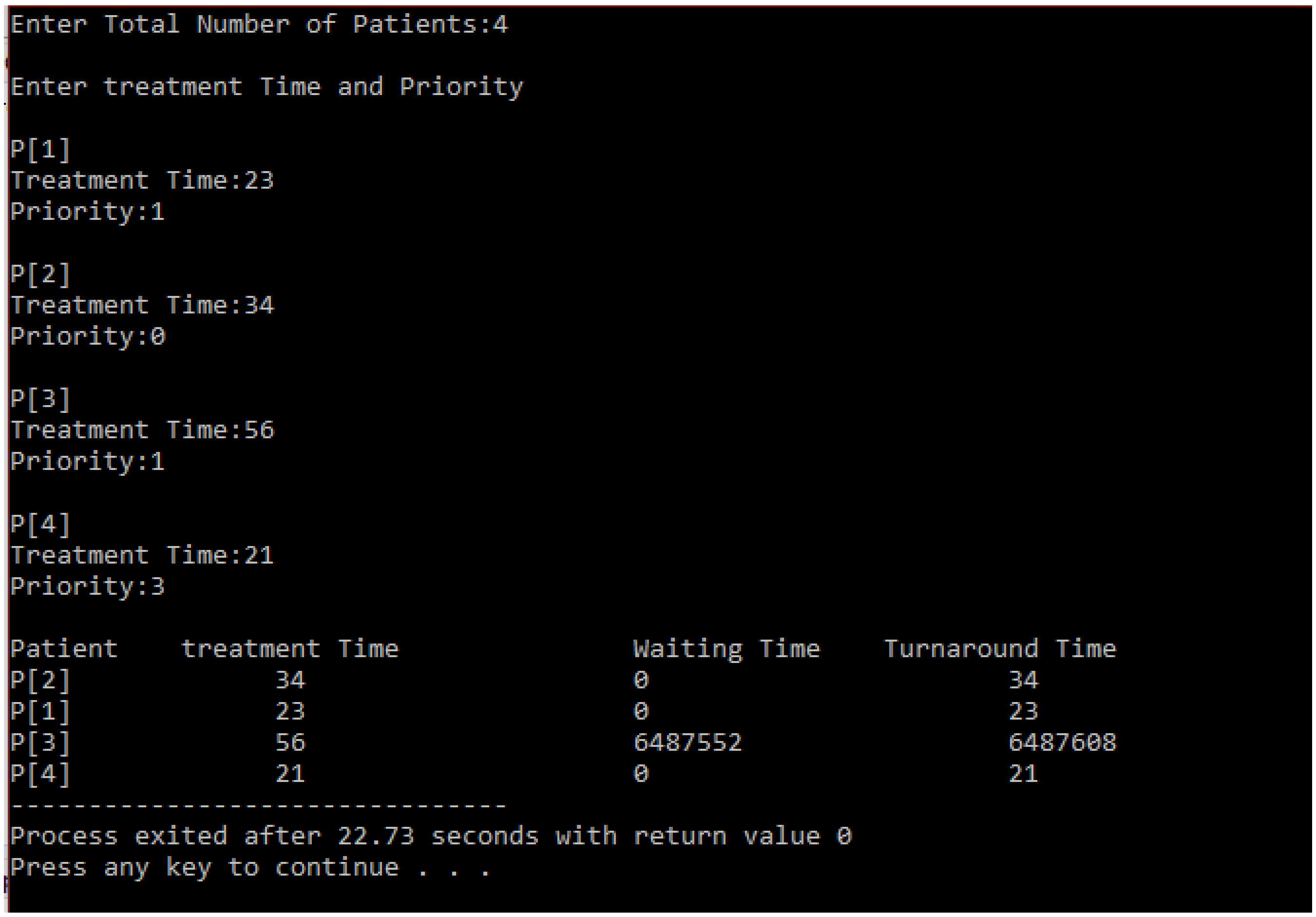
3)Priority and FCFS

FCFS gives the highest priority to the job having been in existence the longest.

PRIORITY:

```
#include<stdio.h>

int main()
{
   int bt[20],p[20],wt[20],tat[20],pr[20],i,j,n,total=0,pos,temp,avg_wt,avg_tat;
   printf("Enter Total Number of Patients:");
   scanf("%d",&n);

   printf("\nEnter treatment Time and Priority\n");
   for(i=0;i<n;i++)
   {
      printf("\nP[%d]\n",i+1);
      printf("Treatment
      Time:");
      scanf("%d",&bt[i]);
      printf("Priority:");
      scanf("%d",&pr[i]);
      p[i]=i+1;        //contains patient number
   }

   //sorting treatment time, priority and patient number in ascending order using
selection sort
   for(i=0;i<n;i++)
   {
      pos=i; for(j=i+1;j<n;j+
      +)
      {
         if(pr[j]<pr[pos])
            pos=j;
      }

      temp=pr[i];
      pr[i]=pr[pos];
      pr[pos]=temp;

      temp=bt[i];
      bt[i]=bt[pos];
      bt[pos]=temp;

      temp=p[i];
      p[i]=p[pos];
      p[pos]=temp;
   }
```

```
    wt[0]=0;   //waiting time for first process is zero

    //calculate waiting
    time for(i=1;i<n;i++)
    {
       wt[i]=0;
       for(j=0;j<i;j++)
          wt[i]+=bt[j];

       total+=wt[i];
    }

    avg_wt=total/n;    //average waiting time
    total=0;

    printf("\nPatient\t   treatment Time    \tWaiting Time\tTurnaround
    Time"); for(i=0;i<n;i++)
    {
       tat[i]=bt[i]+wt[i];     //calculate turnaround time
       total+=tat[i];
       printf("\nP[%d]\t\t %d\t\t   %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }

    avg_tat=total/n;   //average turnaround time
    printf("\n\nAverage Waiting Time=%d",avg_wt);
    printf("\nAverage Turnaround Time=%d\n",avg_tat);

    return 0;
 }
```

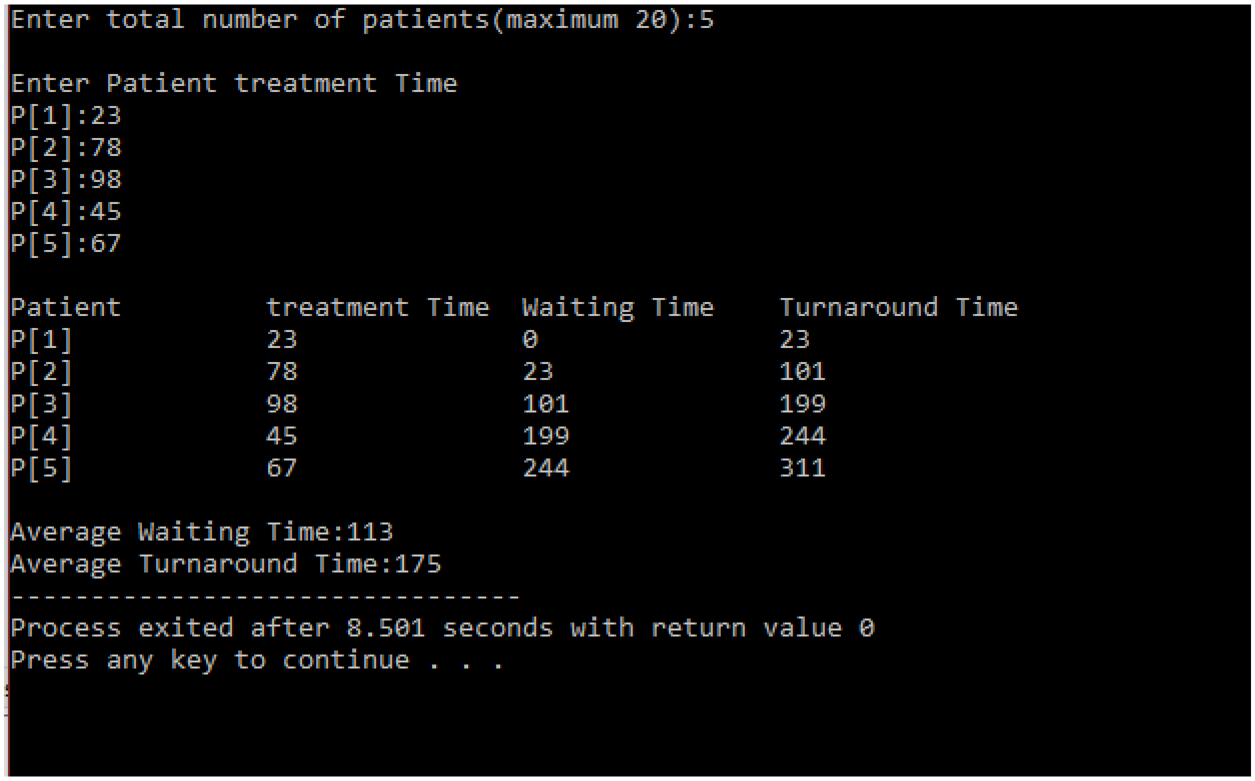


```
Enter Total Number of Patients:4

Enter treatment Time and Priority

P[1]
Treatment Time:23
Priority:1

P[2]
Treatment Time:34
Priority:0

P[3]
Treatment Time:56
Priority:1

P[4]
Treatment Time:21
Priority:3

Patient    treatment Time               Waiting Time    Turnaround Time
P[2]            34                      0                       34
P[1]            23                      0                       23
P[3]            56                      6487552                 6487608
P[4]            21                      0                       21
--------------------------------
Process exited after 22.73 seconds with return value 0
Press any key to continue . . .
```

FCFC:

```
#include<stdio.h>

int main()
{
   int n,bt[20],wt[20],tat[20],avwt=0,avtat=0,i,j;
   printf("Enter total number of patients(maximum 20):");
   scanf("%d",&n);

   printf("\nEnter Patient treatment
   Time\n"); for(i=0;i<n;i++)
   {
      printf("P[%d]:",i+1);
      scanf("%d",&bt[i]);
   }
```

```
    wt[0]=0;   //waiting time for first process is 0


    //calculating waiting time
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
            wt[i]+=bt[j];
    }

    printf("\nPatient\t\ttreatment Time\tWaiting Time\tTurnaround Time");

    //calculating turnaround time
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        avwt+=wt[i];
        avtat+=tat[i];
        printf("\nP[%d]\t\t%d\t\t%d\t\t%d",i+1,bt[i],wt[i],tat[i]);
    }

    avwt/=i;
    avtat/=i;
    printf("\n\nAverage Waiting Time:%d",avwt);
    printf("\nAverage Turnaround Time:%d",avtat);

    return 0;
}
```



```
Enter total number of patients(maximum 20):5

Enter Patient treatment Time
P[1]:23
P[2]:78
P[3]:98
P[4]:45
P[5]:67

Patient         treatment Time  Waiting Time    Turnaround Time
P[1]            23              0               23
P[2]            78              23              101
P[3]            98              101             199
P[4]            45              199             244
P[5]            67              244             311

Average Waiting Time:113
Average Turnaround Time:175
--------------------------------
Process exited after 8.501 seconds with return value 0
Press any key to continue . . .
```

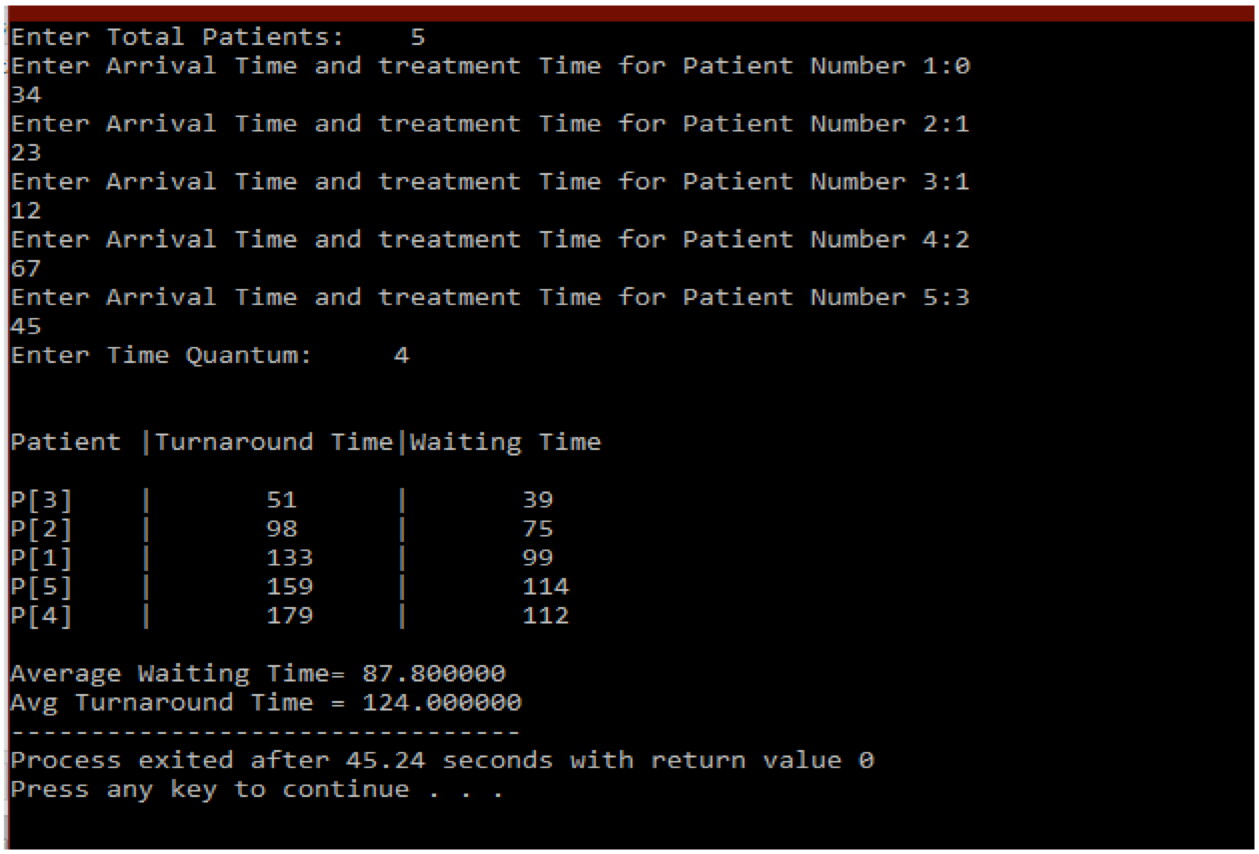
4. RR and SJF

None.

ROUND-ROBIN:

```
#include<stdio.h>

int main()
{

  int count,j,n,time,remain,flag=0,time_quantum;
  int  wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
  printf("Enter Total Patients:\t ");
  scanf("%d",&n);
  remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Arrival Time and treatment Time for Patient Number %d
:",count+1);
    scanf("%d",&at[count]);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
  }
  printf("Enter Time Quantum:\t");
```

```
 scanf("%d",&time_quantum); printf("\n\nPatient\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-at[count],time-at[count]-bt[count]);
   wait_time+=time-at[count]-bt[count];
   turnaround_time+=time-at[count];
   flag=0;
  }
  if(count==n-1)
   count=0;
  else
   if(at[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;
}
```



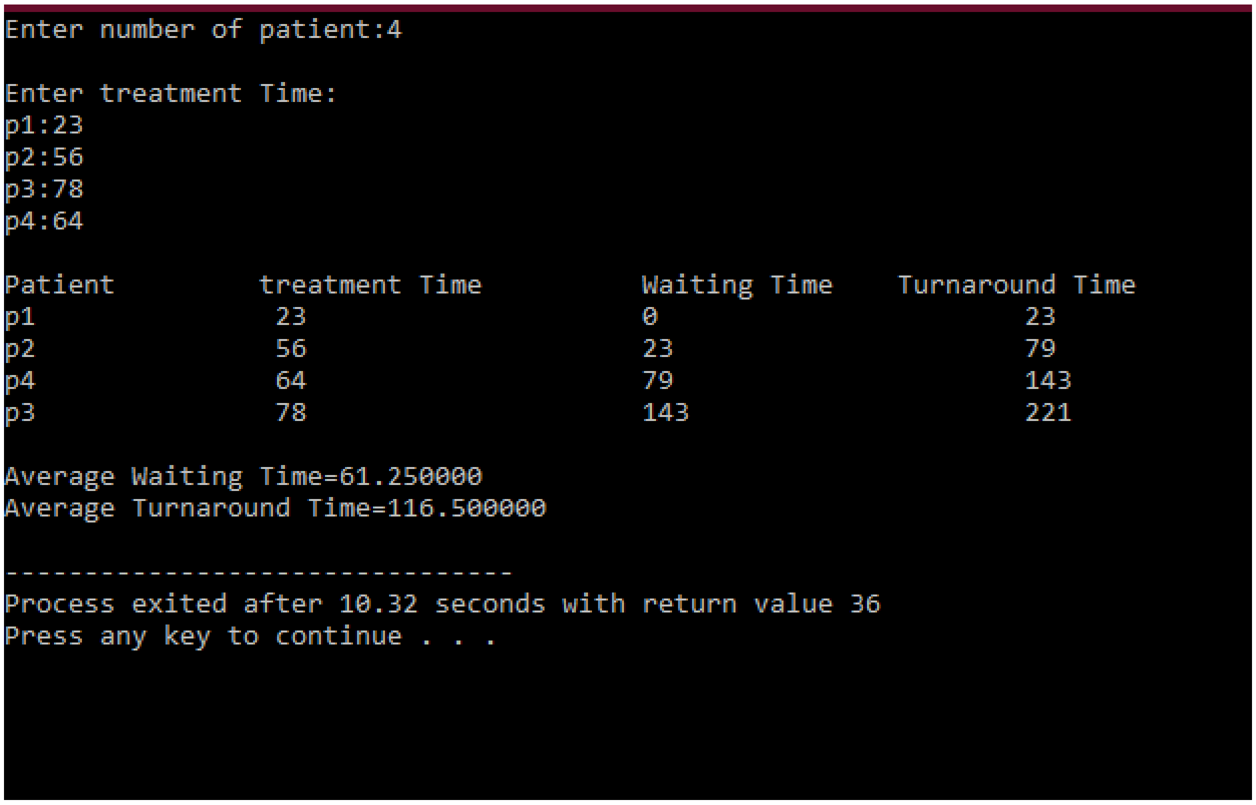
```
Enter Total Patients:    5
Enter Arrival Time and treatment Time for Patient Number 1:0
34
Enter Arrival Time and treatment Time for Patient Number 2:1
23
Enter Arrival Time and treatment Time for Patient Number 3:1
12
Enter Arrival Time and treatment Time for Patient Number 4:2
67
Enter Arrival Time and treatment Time for Patient Number 5:3
45
Enter Time Quantum:     4


Patient |Turnaround Time|Waiting Time

P[3]    |       51      |       39
P[2]    |       98      |       75
P[1]    |       133     |       99
P[5]    |       159     |       114
P[4]    |       179     |       112

Average Waiting Time= 87.800000
Avg Turnaround Time = 124.000000
--------------------------------
Process exited after 45.24 seconds with return value 0
Press any key to continue . . .
```

SJF:

```
#include<stdio.h>

void main()
{
   int  bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;
   float avg_wt,avg_tat;
   printf("Enter number of patient:"); scanf("%d",&n);

   printf("\nEnter treatment Time:\n");
   for(i=0;i<n;i++)
   {
      printf("p%d:",i+1);
      scanf("%d",&bt[i]);
      p[i]=i+1;         //contains patient number
   }

   //sorting burst time in ascending order using selection
```

```
sort for(i=0;i<n;i++)
{
   pos=i; for(j=i+1;j<n;j+
   +)
   {
      if(bt[j]<bt[pos])
         pos=j;
   }

   temp=bt[i];
   bt[i]=bt[pos];
   bt[pos]=temp;

   temp=p[i];
   p[i]=p[pos];
   p[pos]=temp;
}
wt[0]=0;         //waiting time for first process will be zero

//calculate waiting
time for(i=1;i<n;i++)
{
   wt[i]=0;
   for(j=0;j<i;j++)
      wt[i]+=bt[j];

   total+=wt[i];
}

avg_wt=(float)total/n;    //average waiting time
total=0;

printf("\nPatient\t    treatment Time   \tWaiting Time\tTurnaround
Time"); for(i=0;i<n;i++)
{
   tat[i]=bt[i]+wt[i];     //calculate turnaround time
   total+=tat[i];
   printf("\np%d\t\t %d\t\t  %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
}

avg_tat=(float)total/n;   //average turnaround time
```

```
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f\n",avg_tat);
}
```



```
Enter number of patient:4

Enter treatment Time:
p1:23
p2:56
p3:78
p4:64

Patient         treatment Time          Waiting Time    Turnaround Time
p1               23                     0                       23
p2               56                     23                      79
p4               64                     79                      143
p3               78                     143                     221

Average Waiting Time=61.250000
Average Turnaround Time=116.500000

--------------------------------
Process exited after 10.32 seconds with return value 36
Press any key to continue . . .
```