CSE 2005 OPERATING SYSTEM

L51+52

ASSIGNMENT - II

19BCE2612

ISHAN KHADKA

CSE2005-Operating Systems Lab

Assessment-2 Questions

CPU Scheduling

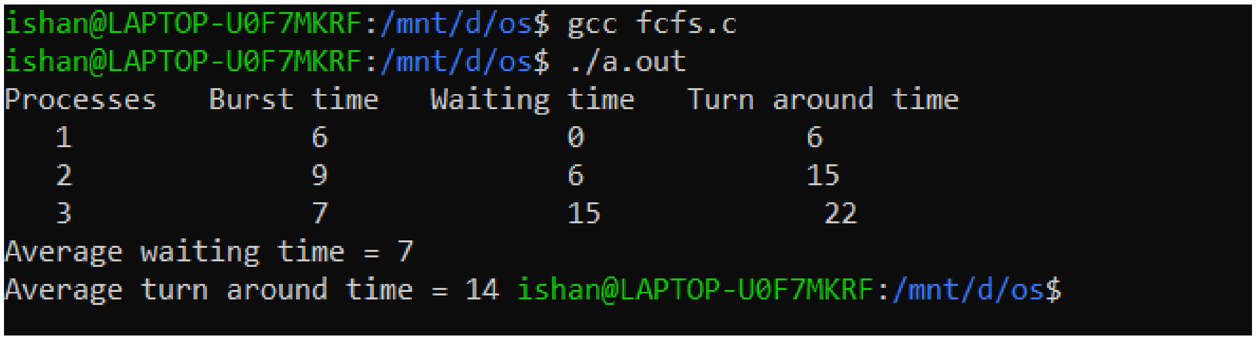
(a) Implement the various process scheduling algorithms such as FCFS, SJF, Priority (Non Preemptive). **(Easy)**

FCFS

```
#include<stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[])

{

   wt[0] = 0;

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

   wt[i] =  bt[i-1] + wt[i-1] ;

}



void findTurnAroundTime( int processes[], int n, int bt[], int wt[], int tat[])

{


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

       tat[i] = bt[i] + wt[i];

}
```

```
void findavgTime( int processes[], int n, int bt[])
{
   int wt[n], tat[n], total_wt = 0, total_tat = 0;
   findWaitingTime(processes, n, bt, wt);
   findTurnAroundTime(processes, n, bt, wt, tat);
   printf("Processes   Burst time   Waiting time   Turn around time\n");
   for (int  i=0; i<n; i++)
   {
       total_wt = total_wt + wt[i];
       total_tat = total_tat + tat[i];
       printf("   %d ",(i+1));
       printf("           %d ", bt[i] );
       printf("           %d",wt[i] );
       printf("           %d\n",tat[i] );
   }
   int s=(float)total_wt / (float)n;
   int t=(float)total_tat / (float)n;
   printf("Average waiting time = %d",s);
   printf("\n");
   printf("Average turn around time = %d ",t);
}


int main()
{
   int processes[] = { 1, 2, 3};

   int n = sizeof processes / sizeof processes[0];

   int  burst_time[] = {6, 9, 7};

   findavgTime(processes, n,  burst_time);

   return 0;

}
```

Output:



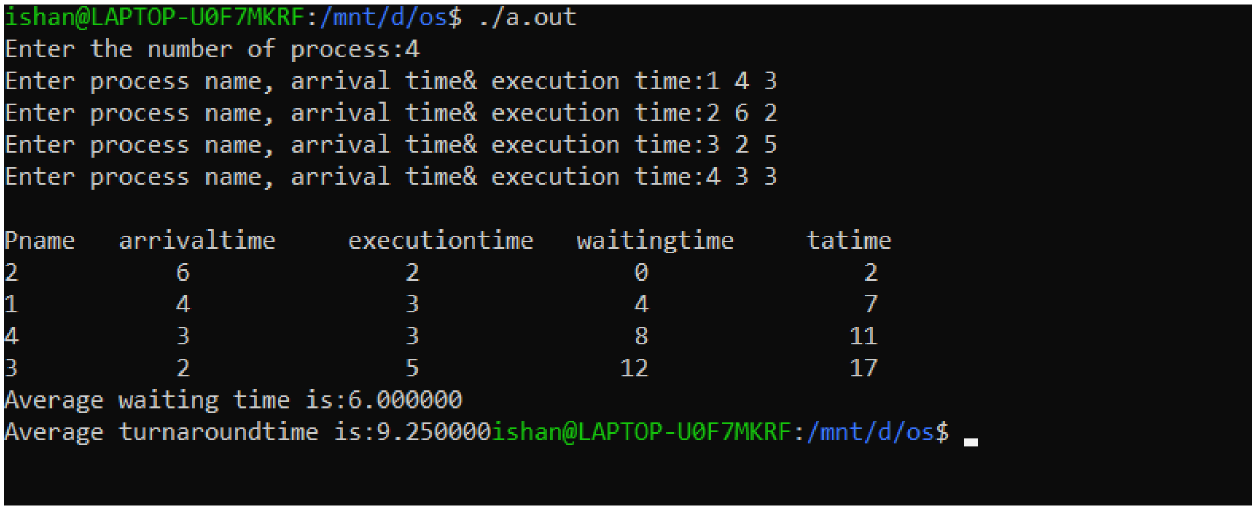
```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc fcfs.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out
Processes    Burst time    Waiting time    Turn around time
   1              6              0              6
   2              9              6              15
   3              7              15              22
Average waiting time = 7
Average turn around time = 14 ishan@LAPTOP-U0F7MKRF:/mnt/d/os$
```

SJF

```
#include<stdio.h>

#include<string.h>

void main()

{

   int et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];

   int totwt=0,totta=0;

   float awt,ata;

   char pn[10][10],t[10];


   printf("Enter the number of process:");
```

```
scanf("%d",&n);
for(i=0; i<n; i++)
{
   printf("Enter process name, arrival time& execution time:");

   scanf("%s%d%d",pn[i],&at[i],&et[i]);
}
for(i=0; i<n; i++)
   for(j=0; j<n; j++)
   {
      if(et[i]<et[j])
      {
         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];
        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\twaitingtime\ttatime");
    for(i=0; i<n; i++)
    printf("\n%s\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],ta[i]);
    printf("\nAverage waiting time is:%f",awt);
    printf("\nAverage turnaroundtime is:%f",ata);

}
```

Output



```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out
Enter the number of process:4
Enter process name, arrival time& execution time:1 4 3
Enter process name, arrival time& execution time:2 6 2
Enter process name, arrival time& execution time:3 2 5
Enter process name, arrival time& execution time:4 3 3

Pname   arrivaltime     executiontime   waitingtime     tatime
2           6               2               0               2
1           4               3               4               7
4           3               3               8              11
3           2               5              12              17
Average waiting time is:6.000000
Average turnaroundtime is:9.250000ishan@LAPTOP-U0F7MKRF:/mnt/d/os$
```

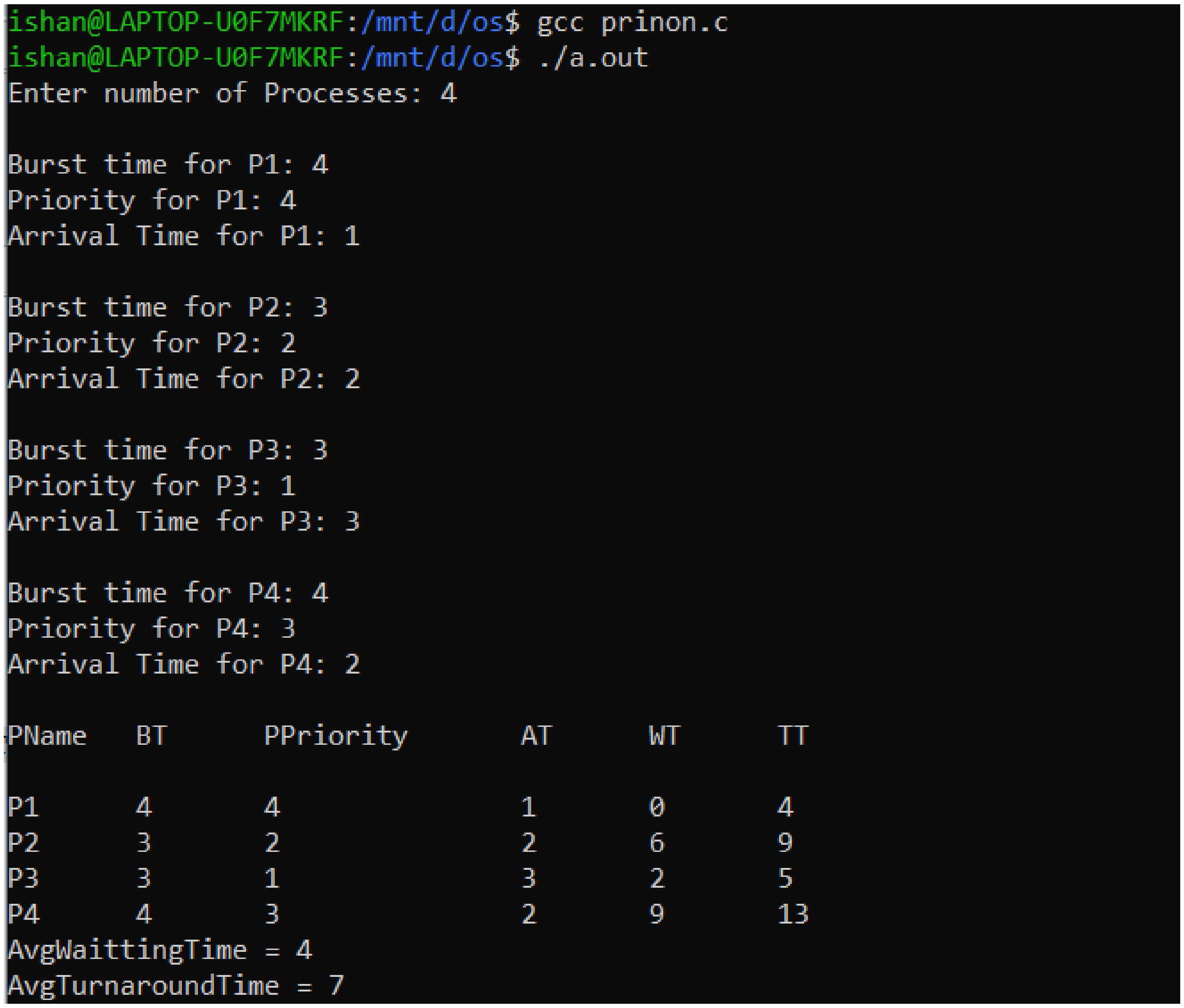
PRIORITY

```
#include <stdlib.h>

#include <stdio.h>




void main()

{


   int pn = 0;

   int CPU = 0;

   int allTime = 0;

   printf("Enter number of Processes: ");

   scanf("%d",&pn);

   int AT[pn];

   int ATt[pn];

   int NoP = pn;
int PT[pn];

int PP[pn];

int PPt[pn];

int waittingTime[pn];

int turnaroundTime[pn];

int i=0;


for(i=0 ;i<pn ;i++){

   printf("\nBurst time for P%d: ",i+1);

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

   printf("Priority for P%d: ",i+1);

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

   PPt[i] = PP[i];

   printf("Arrival Time for P%d: ",i+1);

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

   ATt[i] = AT[i];

}





int LAT = 0;

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

   if(AT[i] > LAT)

      LAT = AT[i];
```

```
int MAX_P = 0;

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

    if(PPt[i] > MAX_P)

        MAX_P = PPt[i];





int ATi = 0;

int P1 = PPt[0];

int P2 = PPt[0];




int j = -1;

while(NoP > 0 && CPU <= 1000){

    for(i = 0; i < pn; i++){

        if((ATt[i] <= CPU) && (ATt[i] != (LAT+10))){

            if(PPt[i] != (MAX_P+1)){

                P2 = PPt[i];

                j= 1;


                if(P2 < P1){

                    j= 1;
                ATi = i;

                P1 = PPt[i];

                P2 = PPt[i];

            }

        }

    }

}


if(j == -1){

    CPU = CPU+1;

    continue;

}else{



    waittingTime[ATi] = CPU - ATt[ATi];

    CPU = CPU + PT[ATi];

    turnaroundTime[ATi] = CPU - ATt[ATi];

    ATt[ATi] = LAT +10;

    j = -1;

    PPt[ATi] = MAX_P + 1;

    ATi = 0;

    P1 = MAX_P+1;

    P2 = MAX_P+1;

    NoP = NoP - 1;
```

```
        }





    }




    printf("\nPName\tBT\tPPriority\tAT\tWT\tTT\n\n");
    for(i = 0; i < pn; i++){
      printf("P%d\t%d\t%d\t\t%d\t%d\t%d\n",i+1,PT[i],PP[i],AT[i],waittingTime[i],turnaroundTime[i]);
    }

    int AvgWT = 0;
    int AVGTaT = 0;
    for(i = 0; i < pn; i++){
        AvgWT = waittingTime[i] + AvgWT;
        AVGTaT = turnaroundTime[i] + AVGTaT;
    }


   printf("AvgWaittingTime = %d\nAvgTurnaroundTime = %d\n",AvgWT/pn,AVGTaT/pn);

}
```

Output:



```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc prinon.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out
Enter number of Processes: 4

Burst time for P1: 4
Priority for P1: 4
Arrival Time for P1: 1

Burst time for P2: 3
Priority for P2: 2
Arrival Time for P2: 2

Burst time for P3: 3
Priority for P3: 1
Arrival Time for P3: 3

Burst time for P4: 4
Priority for P4: 3
Arrival Time for P4: 2

PName   BT      PPriority       AT      WT      TT

P1      4       4               1       0       4
P2      3       2               2       6       9
P3      3       1               3       2       5
P4      4       3               2       9       13
AvgWaittingTime = 4
AvgTurnaroundTime = 7
```

(b) Implement the various process scheduling algorithms such as Priority, Round Robin (preemptive). **(Medium)**

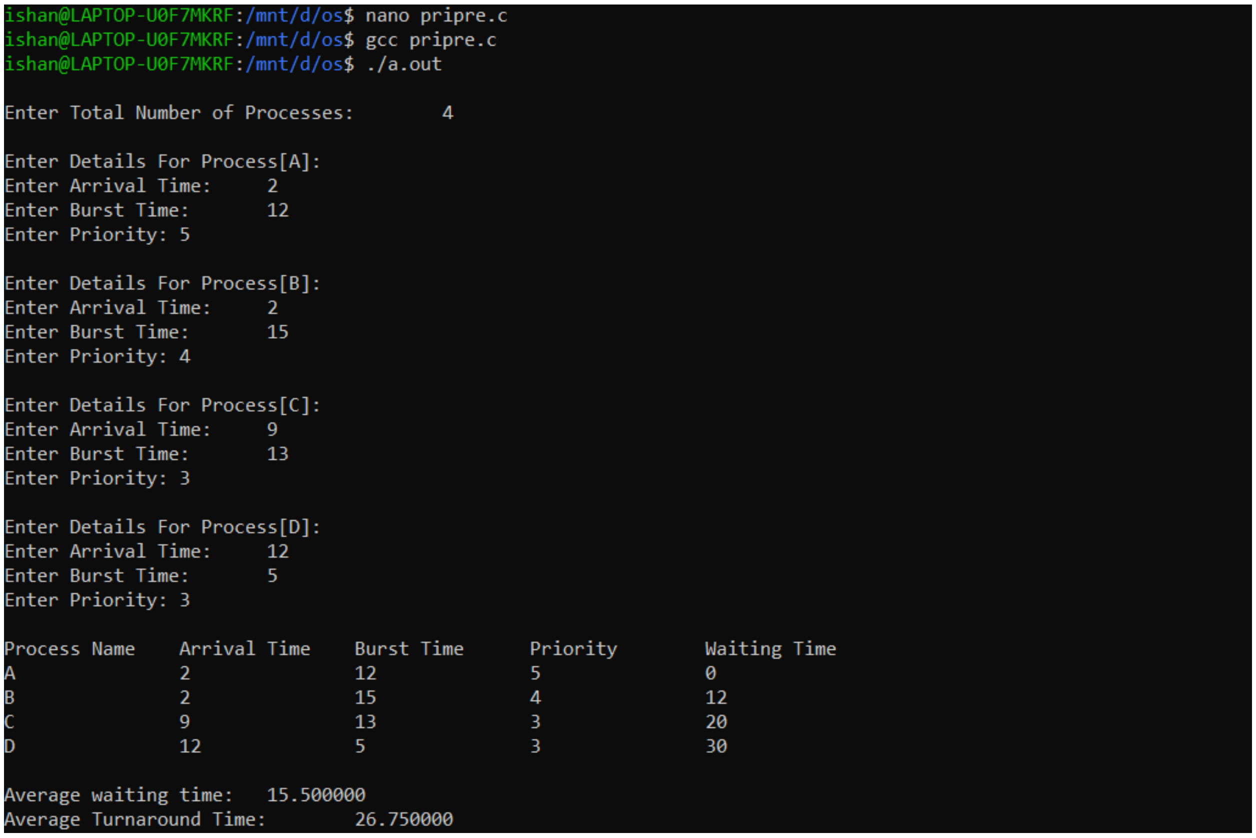
PRIORITY PREEMPTIVE

```
#include<stdio.h>


struct process

{

     char process_name;

     int arrival_time, burst_time, ct, waiting_time, turnaround_time, priority;

     int status;

}process_queue[10];


int limit;


void Arrival_Time_Sorting()

{

     struct process temp;

     int i, j;

     for(i = 0; i < limit - 1; i++)

     {

          for(j = i + 1; j < limit; j++)

          {

               if(process_queue[i].arrival_time > process_queue[j].arrival_time)

               {

                    temp = process_queue[i];
                    process_queue[i] = process_queue[j];
                    process_queue[j] = temp;
                }
            }
        }
    }

void main()
{
    int i, time = 0, burst_time = 0, largest;
    char c;
    float wait_time = 0, turnaround_time = 0, average_waiting_time, average_turnaround_time;
    printf("\nEnter Total Number of Processes:\t");
    scanf("%d", &limit);
    for(i = 0, c = 'A'; i < limit; i++, c++)
    {
        process_queue[i].process_name = c;
        printf("\nEnter Details For Process[%C]:\n", process_queue[i].process_name);
        printf("Enter Arrival Time:\t");
        scanf("%d", &process_queue[i].arrival_time );
        printf("Enter Burst Time:\t");
        scanf("%d", &process_queue[i].burst_time);
        printf("Enter Priority:\t");
        scanf("%d", &process_queue[i].priority);
        process_queue[i].status = 0;
```

```
          burst_time = burst_time + process_queue[i].burst_time;
      }
      Arrival_Time_Sorting();
      process_queue[9].priority = -9999;
      printf("\nProcess Name\tArrival Time\tBurst Time\tPriority\tWaiting Time");
      for(time = process_queue[0].arrival_time; time < burst_time;)
      {
          largest = 9;
          for(i = 0; i < limit; i++)
          {
              if(process_queue[i].arrival_time <= time && process_queue[i].status != 1 &&
  process_queue[i].priority > process_queue[largest].priority)
              {
                  largest = i;
              }
          }
          time = time + process_queue[largest].burst_time;
          process_queue[largest].ct = time;
          process_queue[largest].waiting_time = process_queue[largest].ct -
  process_queue[largest].arrival_time - process_queue[largest].burst_time;
          process_queue[largest].turnaround_time = process_queue[largest].ct -
  process_queue[largest].arrival_time;
          process_queue[largest].status = 1;
          wait_time = wait_time + process_queue[largest].waiting_time;
          turnaround_time = turnaround_time + process_queue[largest].turnaround_time;
```

```
            printf("\n%c\t\t%d\t\t%d\t\t%d\t\t%d", process_queue[largest].process_name,
process_queue[largest].arrival_time, process_queue[largest].burst_time,
process_queue[largest].priority, process_queue[largest].waiting_time);

      }

      average_waiting_time = wait_time / limit;

      average_turnaround_time = turnaround_time / limit;

      printf("\n\nAverage waiting time:\t%f\n", average_waiting_time);

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

}
```

Output:



```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ nano pripre.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc pripre.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out

Enter Total Number of Processes:        4

Enter Details For Process[A]:
Enter Arrival Time:     2
Enter Burst Time:       12
Enter Priority: 5

Enter Details For Process[B]:
Enter Arrival Time:     2
Enter Burst Time:       15
Enter Priority: 4

Enter Details For Process[C]:
Enter Arrival Time:     9
Enter Burst Time:       13
Enter Priority: 3

Enter Details For Process[D]:
Enter Arrival Time:     12
Enter Burst Time:       5
Enter Priority: 3

Process Name    Arrival Time    Burst Time      Priority        Waiting Time
A               2               12              5               0
B               2               15              4               12
C               9               13              3               20
D               12              5               3               30

Average waiting time:   15.500000
Average Turnaround Time:        26.750000
```

ROUND ROBIN

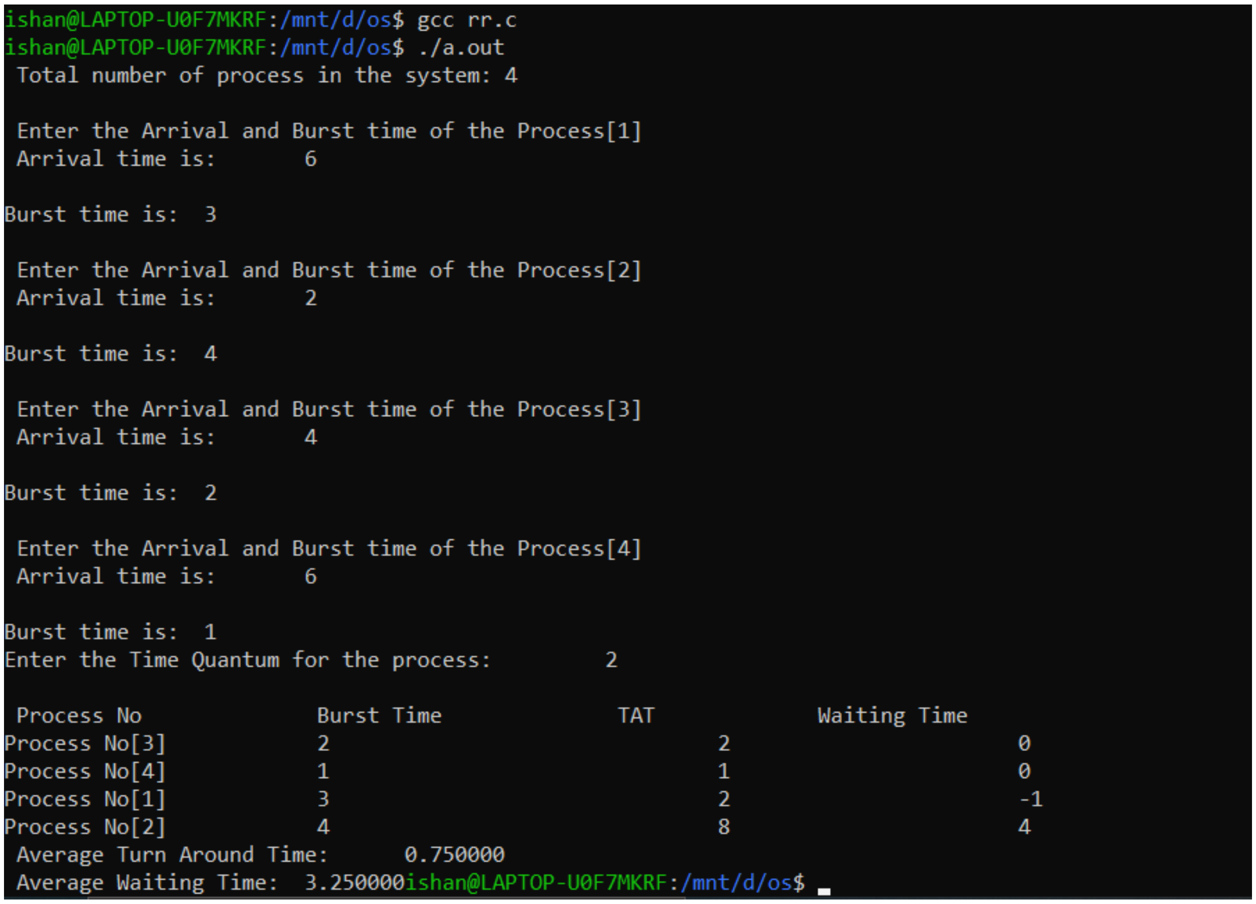
```
#include<stdio.h>

void main()
```

```
{


   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];

   float avg_wt, avg_tat;

   printf(" Total number of process in the system: ");

   scanf("%d", &NOP);

   y = NOP;


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

{

printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);

printf(" Arrival time is: \t");

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

printf(" \nBurst time is: \t");

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

temp[i] = bt[i];

}


printf("Enter the Time Quantum for the process: \t");

scanf("%d", &quant);


printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");

for(sum=0, i = 0; y!=0; )

{

if(temp[i] <= quant && temp[i] > 0)
```

```
{
   sum = sum + temp[i];
   temp[i] = 0;
   count=1;
   }
   else if(temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if(temp[i]==0 && count==1)
   {
      y--;
      printf("\nProcess No[%d] \t\t %d\t\t\t\t %d\t\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
      wt = wt+sum-at[i]-bt[i];
      tat = tat+sum-at[i];
      count =0;
   }
   if(i==NOP-1)
   {
      i=0;
   }
   else if(at[i+1]<=sum)
   {
      i++;
```

```
    }
    else
    {
        i=0;
    }
}
avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
}
```

Output:



```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc rr.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out
 Total number of process in the system: 4

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       6

Burst time is:  3

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       2

Burst time is:  4

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       4

Burst time is:  2

 Enter the Arrival and Burst time of the Process[4]
 Arrival time is:       6

Burst time is:  1
Enter the Time Quantum for the process:         2

 Process No             Burst Time              TAT             Waiting Time
Process No[3]           2                               2                       0
Process No[4]           1                               1                       0
Process No[1]           3                               2                       -1
Process No[2]           4                               8                       4
 Average Turn Around Time:      0.750000
 Average Waiting Time:  3.250000ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ _
```

(c) 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 other patients, 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.

1. FCFS

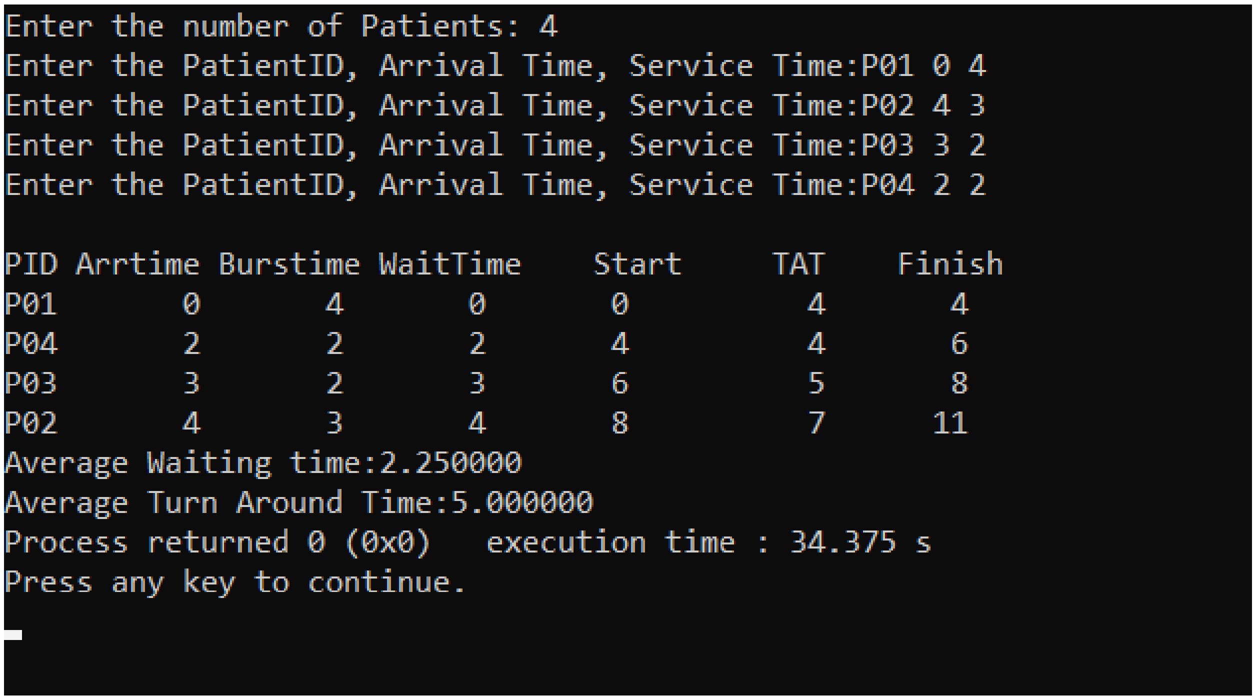
2. SJF (primitive and non-pre-emptive) **(High)**

FCFS

```
#include<stdio.h>

#include<string.h>

int main()

{

   char pn[10][10],t[10];

   int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,j,n,temp;

   int totwt=0,tottat=0;

   printf("Enter the number of Patients:");

   scanf("%d",&n);

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

```
{
    printf("Enter the PatientID, Arrival Time, Service Time:");
    scanf("%s%d%d",&pn[i],&arr[i],&bur[i]);
}
for(i=0; i<n; i++)
{
    for(j=0; j<n; j++)
    {
        if(arr[i]<arr[j])
        {
            temp=arr[i];
            arr[i]=arr[j];
            arr[j]=temp;
            temp=bur[i];
            bur[i]=bur[j];
            bur[j]=temp;
            strcpy(t,pn[i]);
            strcpy(pn[i],pn[j]);
            strcpy(pn[j],t);
        }

    }
}
for(i=0; i<n; i++)
{
        if(i==0)
            star[i]=arr[i];
        else
            star[i]=finish[i-1];
        wt[i]=star[i]-arr[i];
        finish[i]=star[i]+bur[i];
        tat[i]=finish[i]-arr[i];
    }
    printf("\nPID Arrtime Burstime WaitTime      Start        TAT       Finish");
    for(i=0; i<n; i++)
    {
        printf("\n%s\t%3d\t%3d\t%3d\t%3d\t%6d\t%6d",pn[i],arr[i],bur[i],wt[i],star[i],tat[i],finish[i]);
        totwt+=wt[i];
        tottat+=tat[i];
    }
    printf("\nAverage Waiting time:%f",(float)totwt/n);
    printf("\nAverage Turn Around Time:%f",(float)tottat/n);
    return 0;
}
```

Output:



```
Enter the number of Patients: 4
Enter the PatientID, Arrival Time, Service Time:P01 0 4
Enter the PatientID, Arrival Time, Service Time:P02 4 3
Enter the PatientID, Arrival Time, Service Time:P03 3 2
Enter the PatientID, Arrival Time, Service Time:P04 2 2

PID Arrtime Burstime WaitTime    Start     TAT    Finish
P01       0       4       0       0          4       4
P04       2       2       2       4          4       6
P03       3       2       3       6          5       8
P02       4       3       4       8          7      11
Average Waiting time:2.250000
Average Turn Around Time:5.000000
Process returned 0 (0x0)   execution time : 34.375 s
Press any key to continue.
```

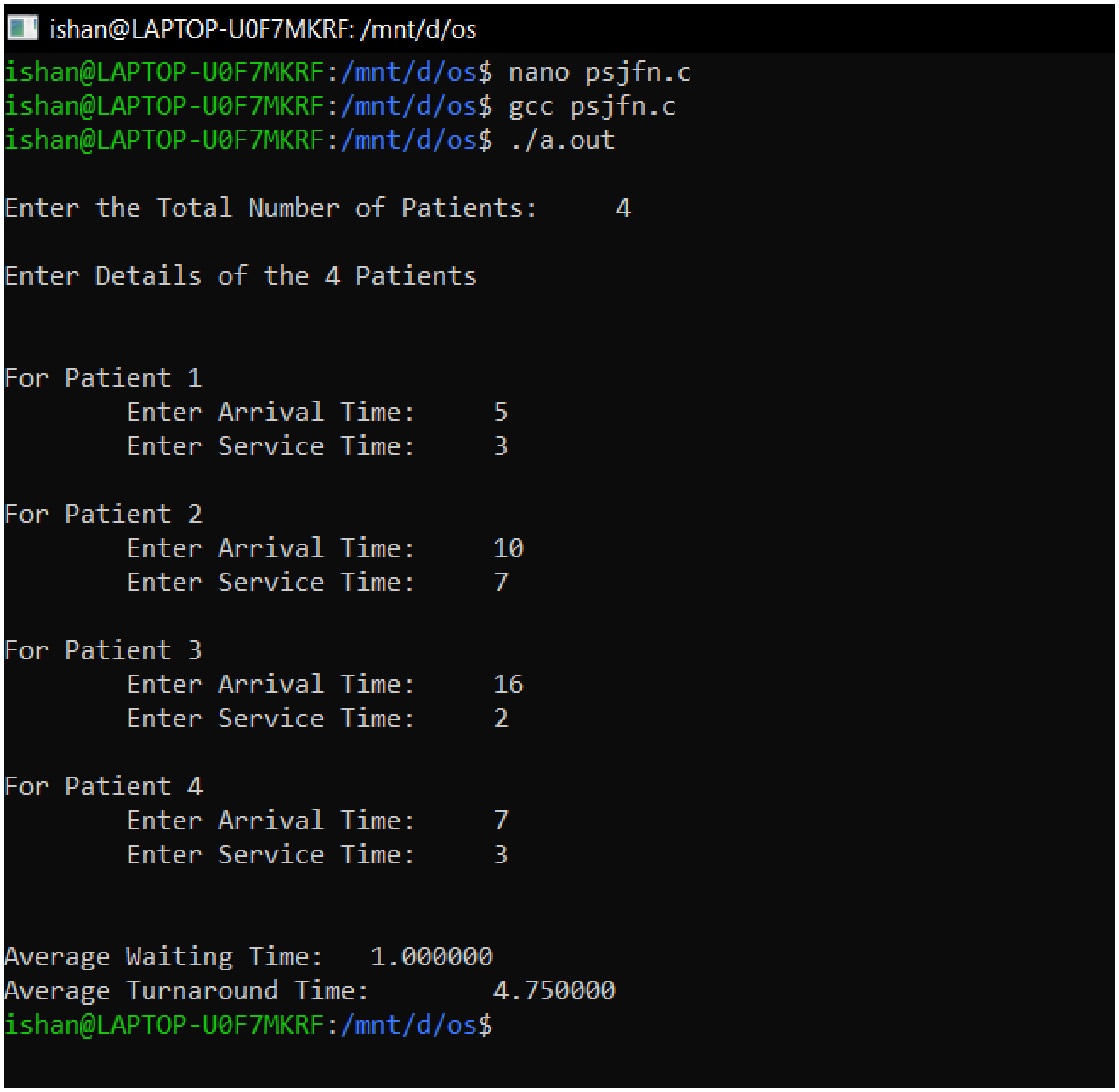
SJF NON PRE EMPTIVE

```
#include <stdio.h>


int main()

{

    int arrival_time[10], service_time[10], temp[10];

    int i, smallest, count = 0, time, limit;

    double wait_time = 0, turnaround_time = 0, end;

    float average_waiting_time, average_turnaround_time;

    printf("\nEnter the Total Number of Patients:\t");

    scanf("%d", &limit);

    printf("\nEnter Details of the %d Patients\n\n", limit);

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

```
{
        printf("\nFor Patient %d",i+1);
    printf("\n\tEnter Arrival Time:\t");
    scanf("%d", &arrival_time[i]);
    printf("\tEnter Service Time:\t");
    scanf("%d", &service_time[i]);
    temp[i] = service_time[i];
}
service_time[9] = 9999;
for(time = 0; count != limit; time++)
{
    smallest = 9;
    for(i = 0; i < limit; i++)
    {
        if(arrival_time[i] <= time && service_time[i] < service_time[smallest] && service_time[i] > 0)
        {
            smallest = i;
        }
    }
    service_time[smallest]--;
    if(service_time[smallest] == 0)
    {
        count++;
        end = time + 1;
        wait_time = wait_time + end - arrival_time[smallest] - temp[smallest];
```

```
                turnaround_time = turnaround_time + end - arrival_time[smallest];
            }
      }
      average_waiting_time = wait_time / limit;
      average_turnaround_time = turnaround_time / limit;
      printf("\n\nAverage Waiting Time:\t%lf\n", average_waiting_time);
      printf("Average Turnaround Time:\t%lf\n", average_turnaround_time);
      return 0;
}
```

Output:



```
ishan@LAPTOP-U0F7MKRF: /mnt/d/os
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ nano psjfn.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc psjfn.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out

Enter the Total Number of Patients:     4

Enter Details of the 4 Patients


For Patient 1
        Enter Arrival Time:     5
        Enter Service Time:     3

For Patient 2
        Enter Arrival Time:     10
        Enter Service Time:     7

For Patient 3
        Enter Arrival Time:     16
        Enter Service Time:     2

For Patient 4
        Enter Arrival Time:     7
        Enter Service Time:     3


Average Waiting Time:   1.000000
Average Turnaround Time:        4.750000
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$
```

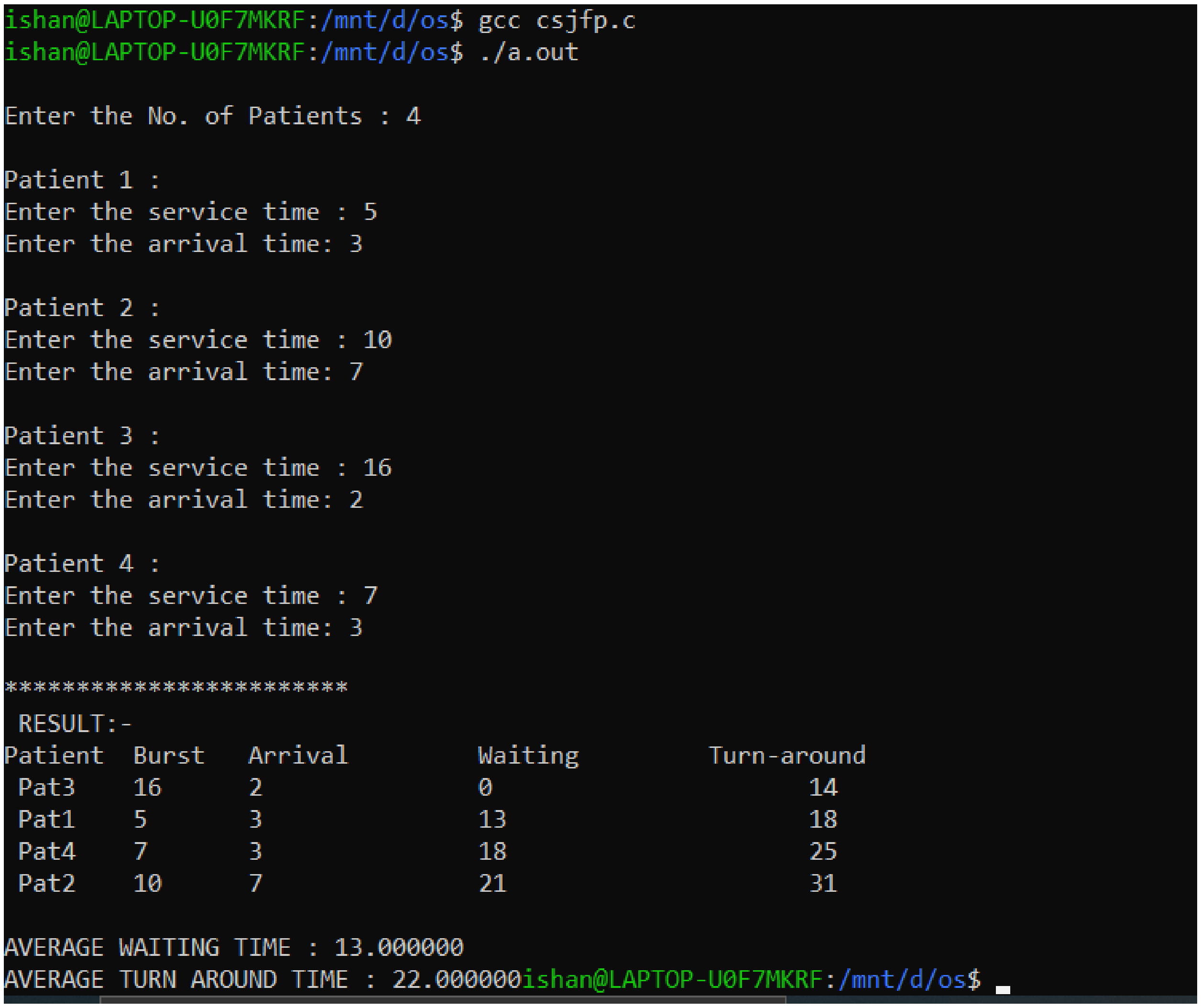
SJF PREEMPTIVE(SRTF)

```
#include<stdio.h>

int main()

{

   int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0;

   int bt[10],temp,j,at[10],wt[10],tt[10],ta=0,sum=0;
```

```
float wavg=0,tavg=0,tsum=0,wsum=0;
printf("\nEnter the No. of Patients : ");
scanf("%d",&n);
printf("\n");

for(i=0;i<n;i++)
{
   printf("Patient %d : \n",i+1);
   printf("Enter the service time : ");
   scanf(" %d",&bt[i]);
   printf("Enter the arrival time: ");
   scanf(" %d",&at[i]);
   printf("\n");
}

for(i=0;i<n;i++)
{
   for(j=0;j<n;j++)
   {
      if(at[i]<at[j])
      {
         temp=p[j];
         p[j]=p[i];
         p[i]=temp;
         temp=at[j];
```

```
            at[j]=at[i];
            at[i]=temp;
            temp=bt[j];
            bt[j]=bt[i];
            bt[i]=temp;
        }
    }
}

for(j=0;j<n;j++)
{
    btime=btime+bt[j];
    min=bt[k];
    for(i=k;i<n;i++)
        {
            if (btime>=at[i] && bt[i]<min)
            {
                temp=p[k];
                p[k]=p[i];
                p[i]=temp;
                temp=at[k];
                at[k]=at[i];
                at[i]=temp;
                temp=bt[k];
                bt[k]=bt[i];
                bt[i]=temp;
        }
    }
    k++;
}
wt[0]=0;
for(i=1;i<n;i++)
{
    sum=sum+bt[i-1];
    wt[i]=sum-at[i];
    wsum=wsum+wt[i];
}

wavg=(wsum/n);
for(i=0;i<n;i++)
{
    ta=ta+bt[i];
    tt[i]=ta-at[i];
    tsum=tsum+tt[i];
}

tavg=(tsum/n);

printf("************************");
printf("\n RESULT:-");
```

```
  printf("\nPatient\t Burst\t Arrival\t Waiting\t Turn-around" );

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

  {

     printf("\n Pat%d\t %d\t %d\t\t %d\t\t\t%d",p[i],bt[i],at[i],wt[i],tt[i]);

  }


  printf("\n\nAVERAGE WAITING TIME : %f",wavg);

  printf("\nAVERAGE TURN AROUND TIME : %f",tavg);

  return 0;

}
```



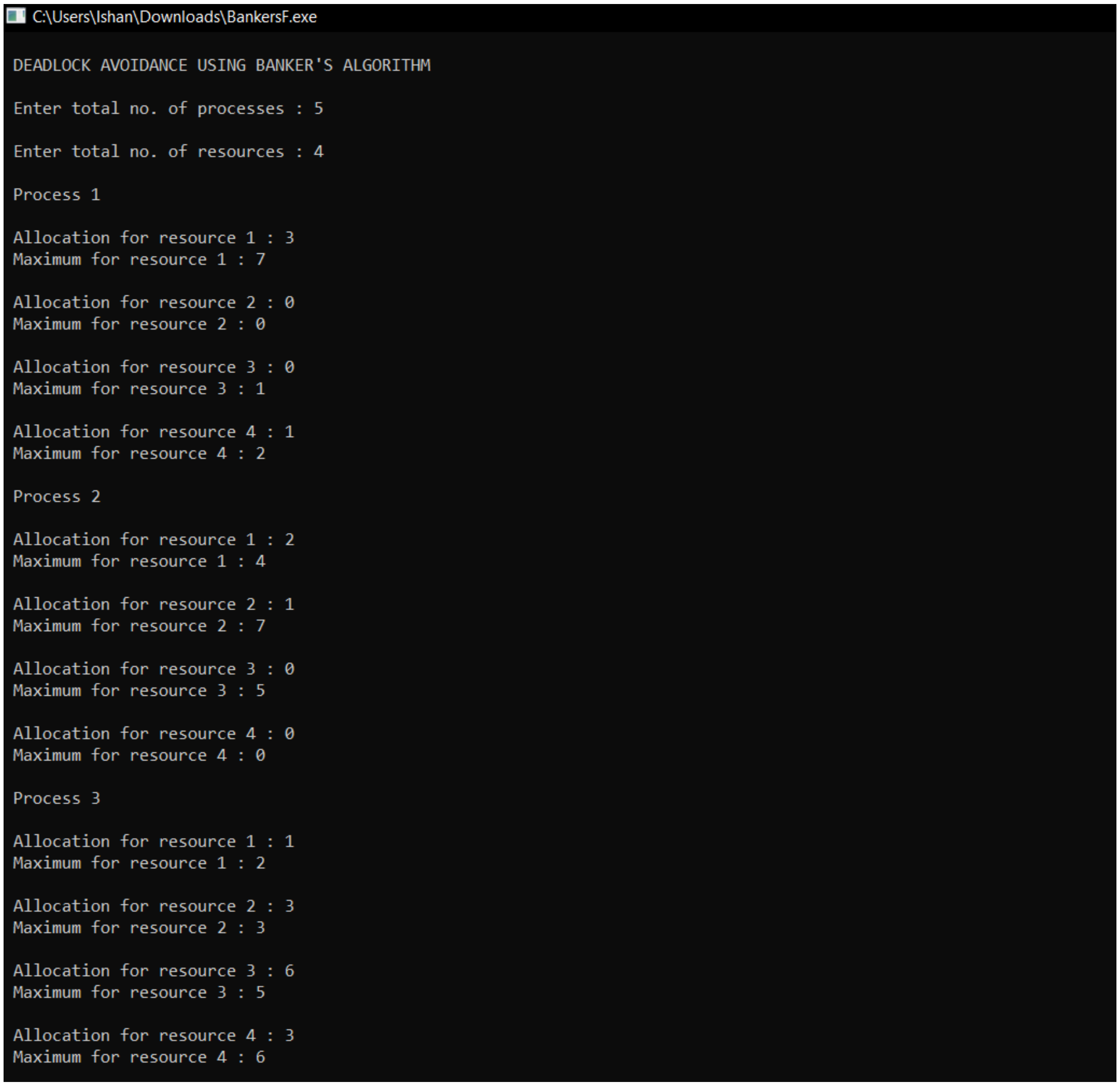
```
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ gcc csjfp.c
ishan@LAPTOP-U0F7MKRF:/mnt/d/os$ ./a.out

Enter the No. of Patients : 4

Patient 1 :
Enter the service time : 5
Enter the arrival time: 3

Patient 2 :
Enter the service time : 10
Enter the arrival time: 7

Patient 3 :
Enter the service time : 16
Enter the arrival time: 2

Patient 4 :
Enter the service time : 7
Enter the arrival time: 3

************************
 RESULT:-
Patient  Burst   Arrival         Waiting         Turn-around
 Pat3    16      2               0                      14
 Pat1    5       3               13                     18
 Pat4    7       3               18                     25
 Pat2    10      7               21                     31

AVERAGE WAITING TIME : 13.000000
AVERAGE TURN AROUND TIME : 22.000000ishan@LAPTOP-U0F7MKRF:/mnt/d/os$
```

(d) Simulate with a program to provide deadlock avoidance of Banker's Algorithm including Safe state and additional resource request **(High)**.

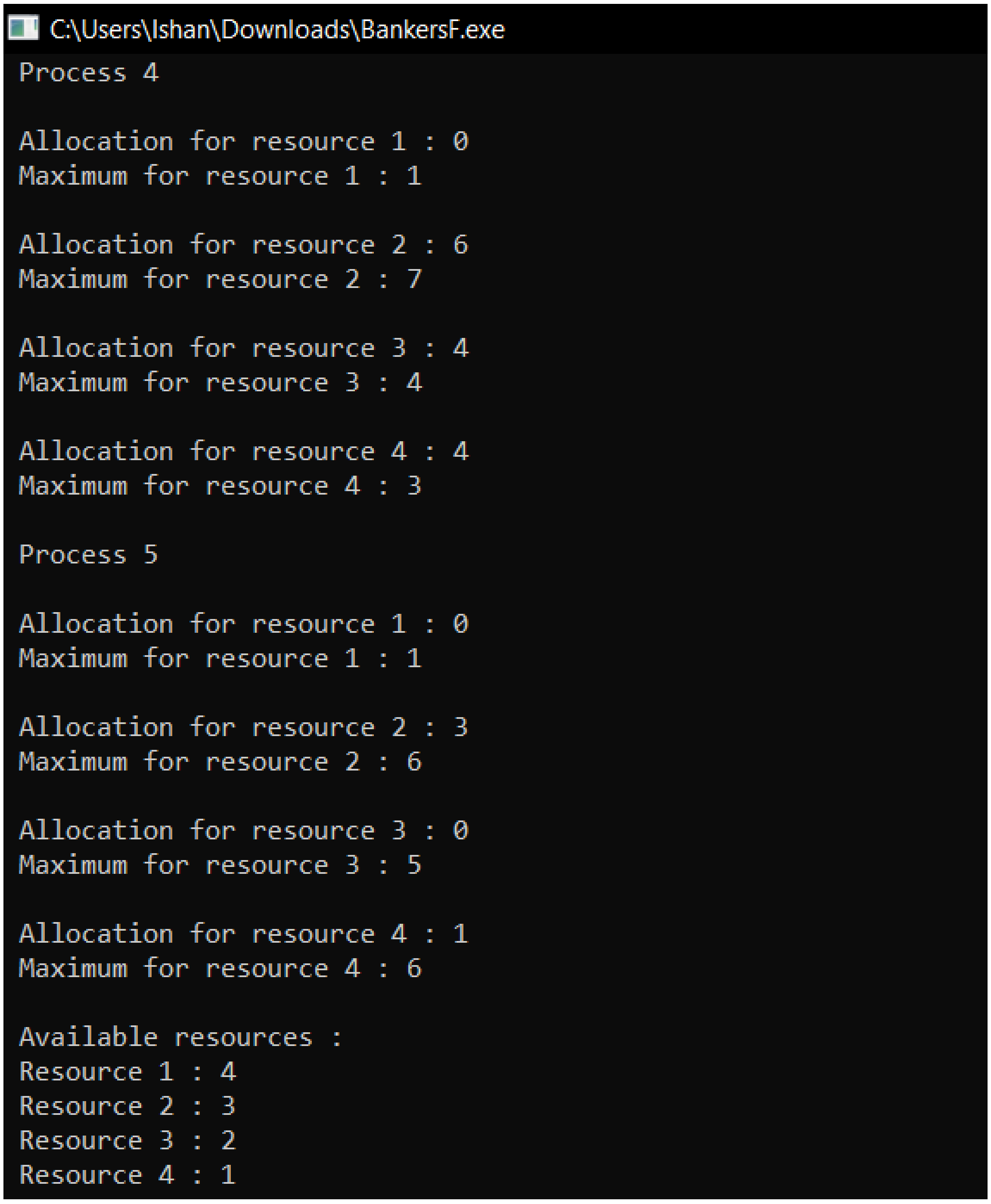
```
#include<stdio.h>

#include<stdlib.h>


void print(int x[][10],int n,int m){

	int i,j;

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

		printf("\n");

		for(j=0;j<m;j++){

			printf("%d\t",x[i][j]);

		}

	}

}



void res_request(int A[10][10],int N[10][10],int AV[10][10],int pid,int m)

{

	int reqmat[1][10];

	int i;

	printf("\n Enter additional request :- \n");

	for(i=0;i<m;i++){

		printf(" Request for resource %d : ",i+1);

		scanf("%d",&reqmat[0][i]);
```

```
        }


        for(i=0;i<m;i++)
                if(reqmat[0][i] > N[pid][i]){
                        printf("\n Error encountered.\n");
                        exit(0);
        }


        for(i=0;i<m;i++)
                if(reqmat[0][i] > AV[0][i]){
                        printf("\n Resources unavailable.\n");
                        exit(0);
                }


        for(i=0;i<m;i++){
                AV[0][i]-=reqmat[0][i];
                A[pid][i]+=reqmat[0][i];
                N[pid][i]-=reqmat[0][i];
        }
}


int safety(int A[][10],int N[][10],int AV[1][10],int n,int m,int a[]){


        int i,j,k,x=0;
        int F[10],W[1][10];
        int pflag=0,flag=0;

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

                F[i]=0;

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

                W[0][i]=AV[0][i];


        for(k=0;k<n;k++){

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

                        if(F[i] == 0){

                                flag=0;

                                for(j=0;j<m;j++){

                                        if(N[i][j] > W[0][j])

                                                flag=1;

                                }

                                if(flag == 0 && F[i] == 0){

                                        for(j=0;j<m;j++)

                                                W[0][j]+=A[i][j];

                                        F[i]=1;

                                        pflag++;

                                        a[x++]=i;

                                }

                        }

                }

                if(pflag == n)

                        return 1;
        }
        return 0;
}


void accept(int A[][10],int N[][10],int M[10][10],int W[1][10],int *n,int *m){
        int i,j;
        printf("\n Enter total no. of processes : ");
        scanf("%d",n);
        printf("\n Enter total no. of resources : ");
        scanf("%d",m);
        for(i=0;i<*n;i++){
                printf("\n Process %d\n",i+1);
                for(j=0;j<*m;j++){
                        printf("\n Allocation for resource %d : ",j+1);
                        scanf("%d",&A[i][j]);
                        printf(" Maximum for resource %d : ",j+1);
                        scanf("%d",&M[i][j]);
                }
        }
        printf("\n Available resources : \n");
        for(i=0;i<*m;i++){
                printf(" Resource %d : ",i+1);
                scanf("%d",&W[0][i]);
        }
        for(i=0;i<*n;i++)
                for(j=0;j<*m;j++)
                        N[i][j]=M[i][j]-A[i][j];

        printf("\n Allocation Matrix");
        print(A,*n,*m);
        printf("\n Maximum Requirement Matrix");
        print(M,*n,*m);
        printf("\n Need Matrix");
        print(N,*n,*m);

}

int banker(int A[][10],int N[][10],int W[1][10],int n,int m){
        int j,i,a[10];
        j=safety(A,N,W,n,m,a);
        if(j != 0 ){
                printf("\n\n");
                for(i=0;i<n;i++)
                   printf(" P%d-> ",a[i]);
                printf("\n A safety sequence has been detected.\n");
                return 1;
        }else{
                printf("\n Deadlock has occured.\n");
                return 0;
        }
}



int main(){
        int ret;
        int A[10][10];
        int M[10][10];
        int N[10][10];
        int W[1][10];
        int n,m,pid,ch;
        printf("\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n");
        accept(A,N,M,W,&n,&m);
        ret=banker(A,N,W,n,m);
        if(ret !=0 ){
                printf("\n Do you want make an additional request ? (1=Yes|0=No)");
                scanf("%d",&ch);
                if(ch == 1){
                        printf("\n Enter process no. : ");
                        scanf("%d",&pid);
                        res_request(A,N,W,pid-1,m);
                        ret=banker(A,N,W,n,m);
                        if(ret == 0 )
                                exit(0);
                }
        }else
                exit(0);
        return 0;
}
```

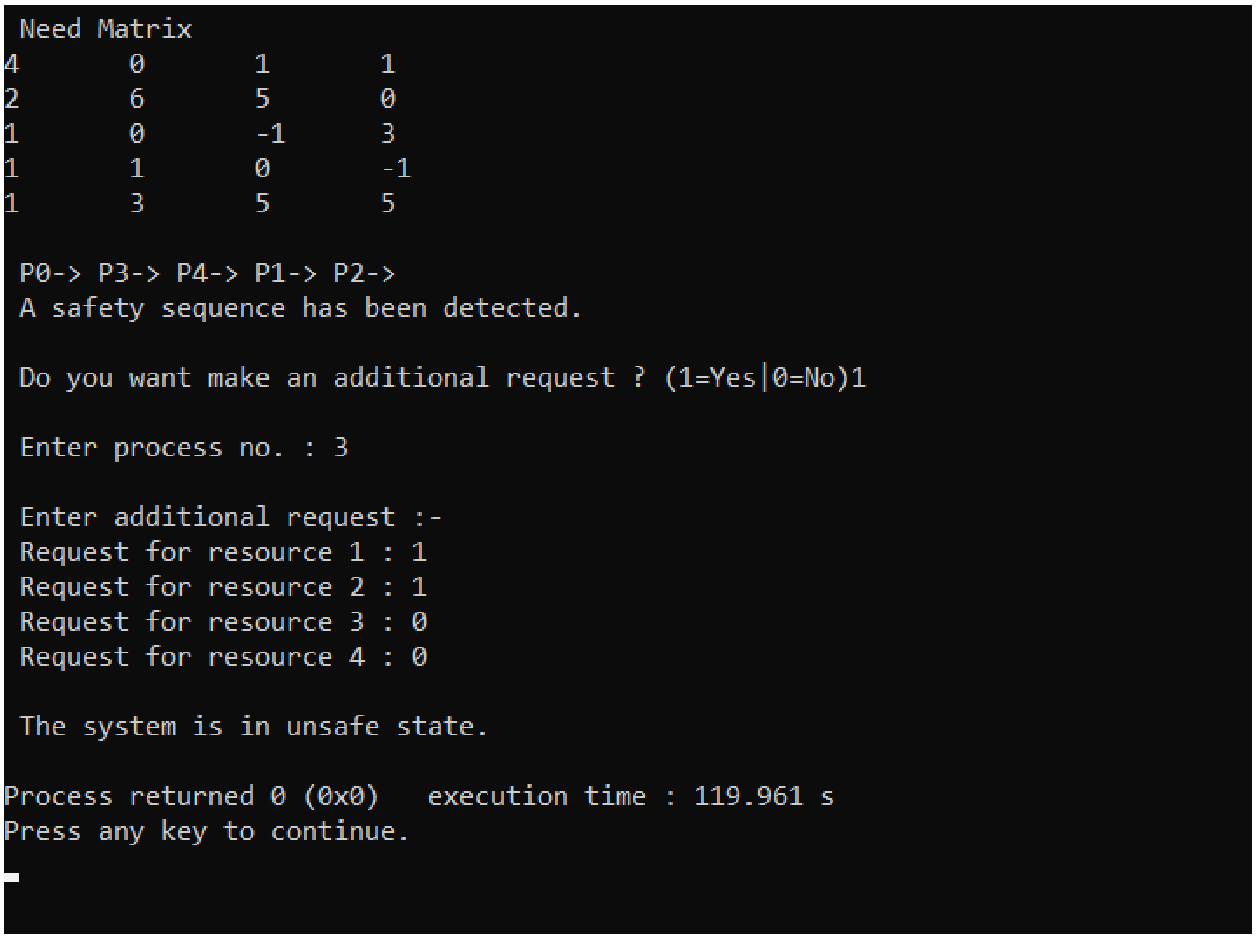
Output


C:\Users\Ishan\Downloads\BankersF.exe

```
DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM

Enter total no. of processes : 5

Enter total no. of resources : 4

Process 1

Allocation for resource 1 : 3
Maximum for resource 1 : 7

Allocation for resource 2 : 0
Maximum for resource 2 : 0

Allocation for resource 3 : 0
Maximum for resource 3 : 1

Allocation for resource 4 : 1
Maximum for resource 4 : 2

Process 2

Allocation for resource 1 : 2
Maximum for resource 1 : 4

Allocation for resource 2 : 1
Maximum for resource 2 : 7

Allocation for resource 3 : 0
Maximum for resource 3 : 5

Allocation for resource 4 : 0
Maximum for resource 4 : 0

Process 3

Allocation for resource 1 : 1
Maximum for resource 1 : 2

Allocation for resource 2 : 3
Maximum for resource 2 : 3

Allocation for resource 3 : 6
Maximum for resource 3 : 5

Allocation for resource 4 : 3
Maximum for resource 4 : 6
```


C:\Users\Ishan\Downloads\BankersF.exe

```
Process 4

Allocation for resource 1 : 0
Maximum for resource 1 : 1

Allocation for resource 2 : 6
Maximum for resource 2 : 7

Allocation for resource 3 : 4
Maximum for resource 3 : 4

Allocation for resource 4 : 4
Maximum for resource 4 : 3

Process 5

Allocation for resource 1 : 0
Maximum for resource 1 : 1

Allocation for resource 2 : 3
Maximum for resource 2 : 6

Allocation for resource 3 : 0
Maximum for resource 3 : 5

Allocation for resource 4 : 1
Maximum for resource 4 : 6

Available resources :
Resource 1 : 4
Resource 2 : 3
Resource 3 : 2
Resource 4 : 1
```



```
 Need Matrix
4       0       1       1
2       6       5       0
1       0       -1      3
1       1       0       -1
1       3       5       5

 P0-> P3-> P4-> P1-> P2->
 A safety sequence has been detected.

 Do you want make an additional request ? (1=Yes|0=No)1

 Enter process no. : 3

 Enter additional request :-
 Request for resource 1 : 1
 Request for resource 2 : 1
 Request for resource 3 : 0
 Request for resource 4 : 0

 The system is in unsafe state.

Process returned 0 (0x0)   execution time : 119.961 s
Press any key to continue.
```