**Student Name:** Satvik Sankrat

**Student Id: 11710417**

**Email Address: satvikegssankrat66@gmail.com**

**GitHub Link:** <https://github.com/iamsatvik/assingment/upload/master>

**Problem 13:** . Considering the arrival time and burst time requirement of the process the scheduler schedules the processes by interrupting the processor after every 6 units of time and does consider the completion of the process in this iteration. The scheduler than checks for the number of process waiting for the processor and allots the processor to the process but interrupting the processor every 10 unit of time and considers the completion of the processes in this iteration.

The scheduler checks the number of processes waiting in the queue for the processor after the second iteration and gives the processor to the process which needs more time to complete than the other processes to go in the terminated state.

The inputs for the number of requirements, arrival time and burst time should be provided by the user.

Consider the following units for reference.

|  |  |  |
| --- | --- | --- |
| Process | Arrival time | Burst time |
| P1 | 0 | 20 |
| P2 | 5 | 36 |
| P3 | 13 | 19 |
| P4 | 26 | 42 |

Develop a scheduler which submits the processes to the processor in the defined scenario, and compute the scheduler performance by providing the waiting time for process, turnaround time for process and average waiting time and turnaround time.

**Code Snippet:**

#include<stdio.h>

#include<conio.h>

void rr(int no,int remt[10],int Cur\_t,int arT[10], int bsT[10]);

main()

{

int Proc\_no,j,no,CurT,RemProc,indicator,time\_quan,wait,tut,arT[10],bsT[10],remt[10],x=1;

indicator = 0;

wait = 0;

tut = 0;

printf("Enter number of processes ");

scanf("%d",&no);

RemProc = no;

printf("\nEnter the arrival time and burst time of the processes\n");

for(Proc\_no = 0;Proc\_no < no;Proc\_no++)

{

printf("\nProcess P%d\n",Proc\_no+1);

printf("Arrival time = ");

scanf("%d",&arT[Proc\_no]);

printf("Burst time = ");

scanf("%d",&bsT[Proc\_no]);

remt[Proc\_no]=bsT[Proc\_no];

}

printf("The details of time quantum are as follows:\n");

printf("The time quantum for first round is 3.\n");

time\_quan=3;

CurT=0;

for(Proc\_no=0;RemProc!=0;)

{

if(remt[Proc\_no]<=time\_quan && remt[Proc\_no]>0)

{

CurT+=remt[Proc\_no];

remt[Proc\_no]=0;

indicator=1;

}

else if(remt[Proc\_no]>0)

{

remt[Proc\_no]-=time\_quan;

CurT+=time\_quan;

}

if(remt[Proc\_no]==0 && indicator==1)

{ printf("%d",Proc\_no);

RemProc--;

printf("P %d",Proc\_no+1);

printf("\t\t\t%d",CurT-arT[Proc\_no]);

printf("\t\t\t%d\n",CurT-bsT[Proc\_no]-arT[Proc\_no]);

wait+=CurT-arT[Proc\_no]-bsT[Proc\_no];

tut+=CurT-arT[Proc\_no];

indicator=0;

}

if(Proc\_no==no-1){

x++;

if(x==2){

Proc\_no=0;

time\_quan=6;

printf("The time quantum for second round is 6. \n");

}

else{

break;

}

}

else if(CurT >= arT[Proc\_no+1]){

Proc\_no++;

}

else{

Proc\_no=0;

}

}

rr(no,remt,CurT,arT,bsT);

return 0;

}

void rr(int no,int remt[10],int Cur\_t,int arT[10], int bsT[10]){

float avg\_wait,avg\_tut;

int i,j,n=no,temp,btime[20],Proc\_no[20],w\_time[20],tut\_t[20],total=0,loc;

printf("Third round with Maximum burst time.\n");

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

{

btime[i]=remt[i];

w\_time[i]=Cur\_t-arT[i]-btime[i];

Proc\_no[i]=i+1;

}

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

{

loc=i;

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

{

if(btime[j]>btime[loc]){

loc=j;

}

}

temp=btime[i];

btime[i]=btime[loc];

btime[loc]=temp;

temp=Proc\_no[i];

Proc\_no[i]=Proc\_no[loc];

Proc\_no[loc]=temp;

}

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

{

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

w\_time[i]+=btime[j];

}

total+=w\_time[i];

}

avg\_wait=(float)total/n;

total=0;

printf("\nProcess\t\tBurst time\t\twaiting time\t\tTurnaround Time");

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

{

tut\_t[i]=btime[i]+w\_time[i];

total=total + tut\_t[i];

printf("\nP%d\t\t\t%d\t\t\t%d\t\t\t%d",Proc\_no[i],btime[i],w\_time[i],tut\_t[i]);

}

avg\_tut=(float)total/n;

printf("\n\nAverage waiting time = %f",avg\_wait);

printf("\n Average turnaround time = %f\n",avg\_tut);

}

**Description:**

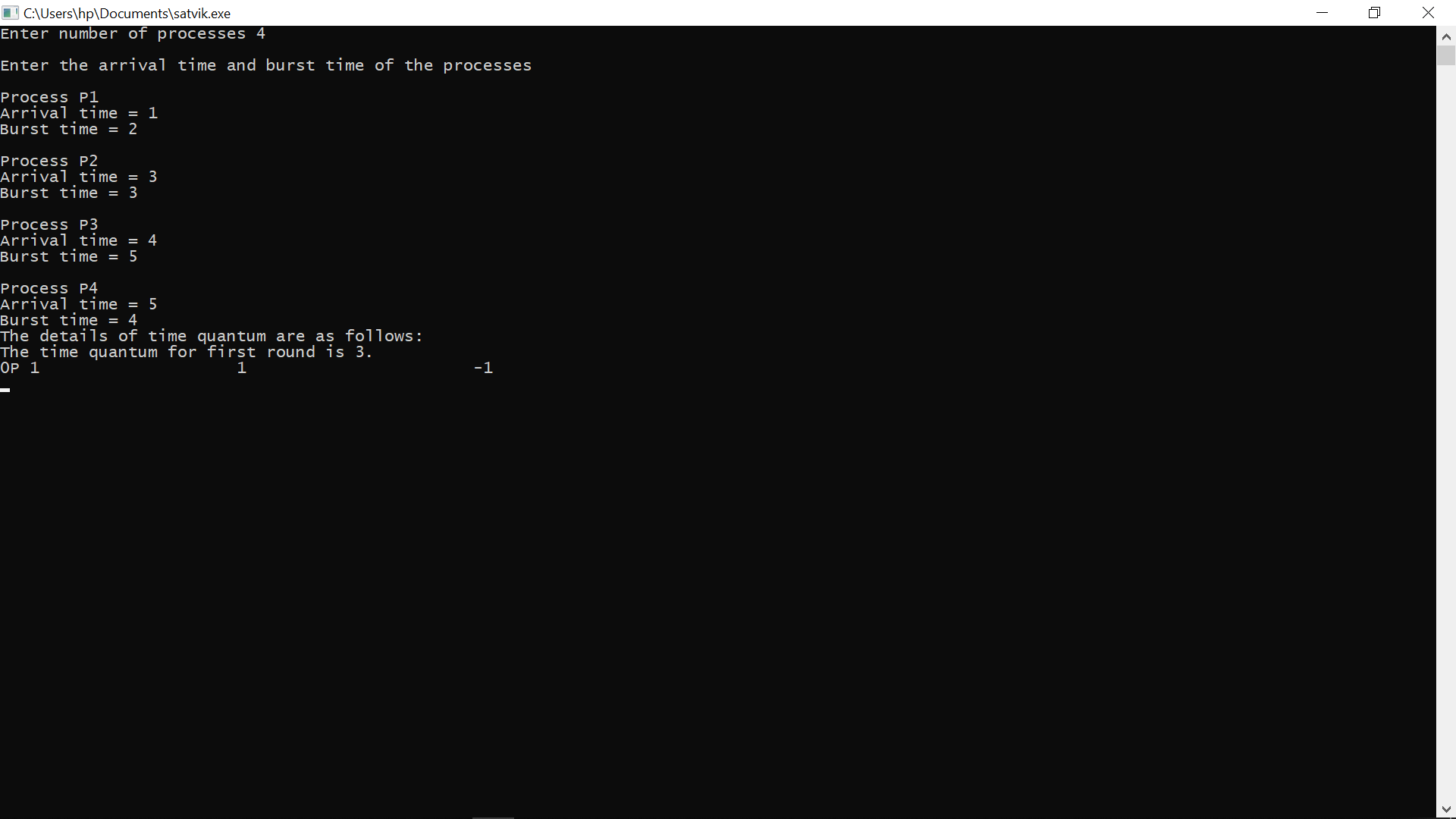
In the above problem, we need to implement a program for multi-level scheduling. It has three queues and on these queues different type of scheduling is applied according to its priority. Every processes arrival time should be 0 else it will generate error.

If the queue has highest priority then it should use round robin scheduling, if it has medium priority then Priority Scheduling is used and for lowest priority FCFS is used. On the three queues Round Robin is used with quantum 10 with its own priority decided scheduling in multilevel scheduling.

**Constraints:**

* Burst time should be greater than 0 and less than 100
* Arrival time should be 0 else it will generate error

**Applied Test cases / Boundary Conditions:**



**GitHub Link:** <https://github.com/iamsatvik/assingment/upload/master>