Name:Sudhanshu Pandey 

Reg:11807636

Email:psudhanshu26@gmail.com

Github link: <https://github.com/Sudhanshu-cloud/operating_system>

Code: question 2

Problem explanation:

Ques. 2. Considering 4 processes with the arrival time and the burst time requirement of the processes the scheduler schedules the processes by interrupting the processor after every 3 units of time and does consider the completion of the process in this iteration. The schedulers then checks for the number of processes waiting for the processor and allots the processor to the process but interrupting the processor after every 6 units of time and considers the completion of the process in this iteration. The scheduler after the second iteration checks for the number of processes waiting for the processor and now provides the processor to the process with the least time requirement 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     18

P2    2     23

P3     4     13

P4     13     10

Develop a scheduler which submits the processes to the processor in the above 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.

Algorithm:

1. For First and Second Iteration.

2. For first iteration, time Quantum =3 and for second Iteration time Quantum =6.

3. For(i=0 to total no of processes): //n times

If(arrival Time < Time Duration):

If(burst Time > time Quantam):{

Decrease burst Time by time Quantum; 

}

Else:{

Time Duration += burst Time;

Compilation Time = Time Duration;

Turn Around Time = Compilation Time – arrival Time;

Burst Time -= time Quantum;

Add the process to the array []; //as to check whether this process is completed later.

}

4. then we Sort remaining processes according to remaining burst Time.

5. then Check for the process which is already completed.

6. For(remaining process): { //(n –k)times

Time Duration += burst Time;

Compilation Time=time Duration;

Turn Around Time = Compilation Time – arrival Time;

}

7. we Print all the process. And Average Turn Around Time and Average Waiting Time.

Complexity: O(n)

Constraints:

Multiple processes cannot enter at the same time i.e their arrival times has to be different

The scheduler has to check after iterations that how much work is completed

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 least 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);

}

Boundry conditions:



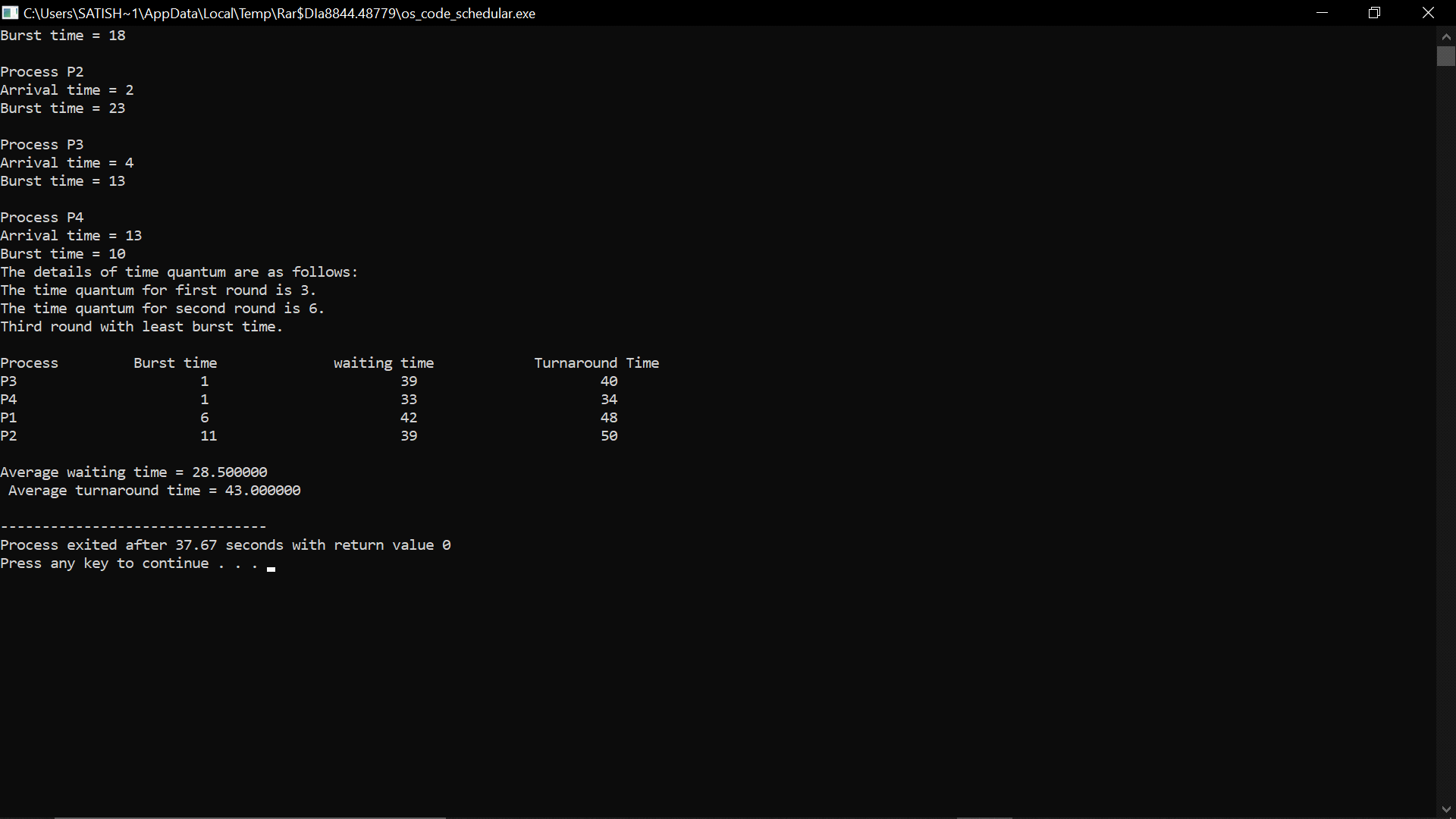
Number of processes cannot be a negative number, it has to be a positive number

Arrival times and burst time also cannot be negative because time cannot be negative

Arrival times and burst time should be in range of integer datatype and not decimal i.e till 32767

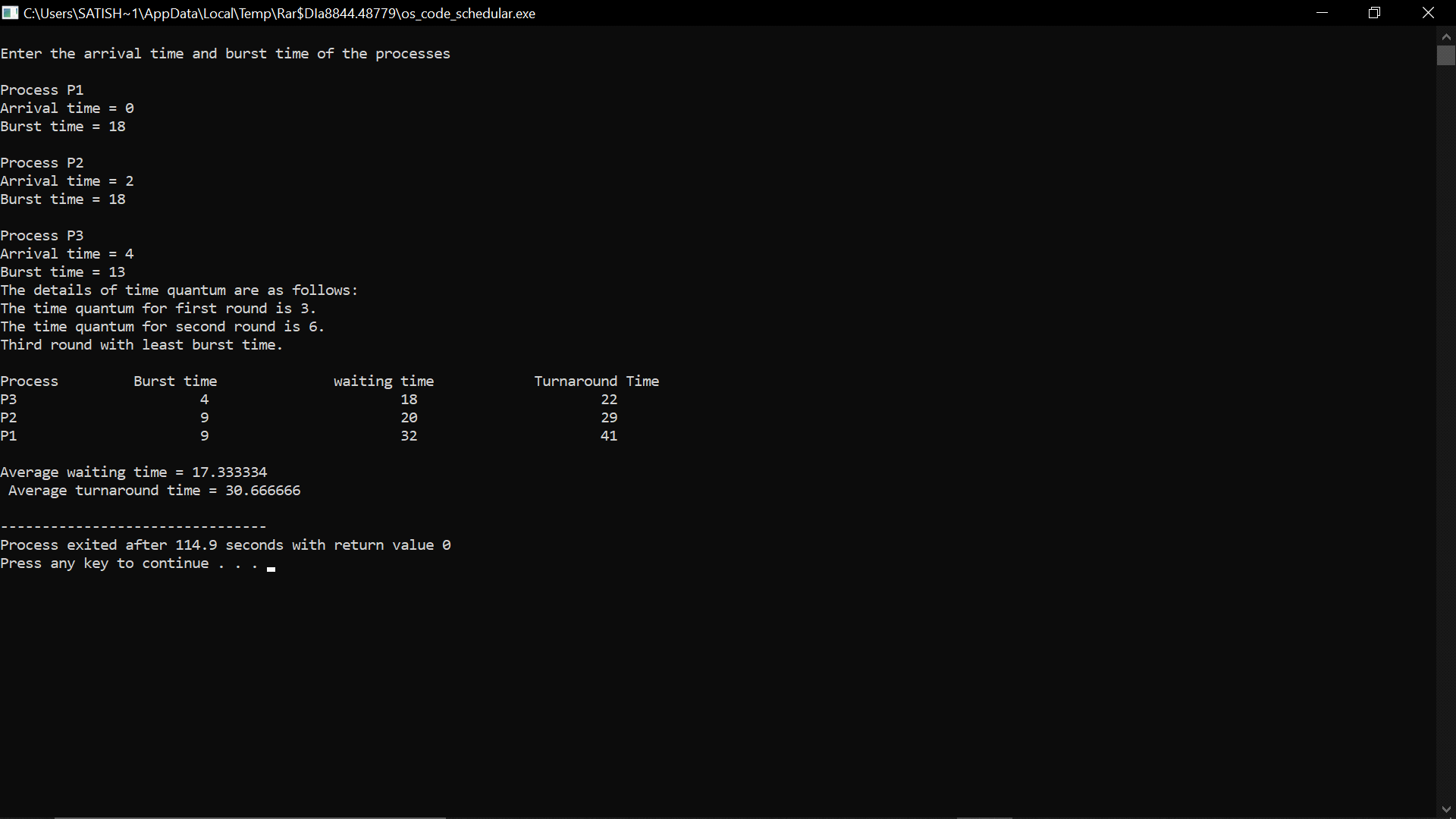
Test cases:

According to entered arrival times and burst times the scheduler first calculates and stores waiting time of all processes and then the turn around time after that the average waiting time and average turn around time is calculated and displayed which is 37.67 and 68.0 respectively.



Explanation: in this image we have entered the data as given in the question first and then after executing for first iteration taking time quantum 3 and then second iteration with time quantum 6 and displayed the stats after 2 iterations in which p1 was complete and we have done third iteration with least burst time and solved accordingly and again displayed the output and average waiting time and average turn around time.





Explanation: in this image we have entered the data by our will and then after executing for first iteration taking time quantum 3 and then second iteration with time quantum 6 and displayed the stats after 2 iterations in which p1 and p3 was complete and we have done third iteration with least burst time and solved accordingly and again displayed the output and average waiting time and average turn around time.