Name:Ashutosh Gupta

Reg:11807745

Email:www.ashutosh221299@gmail.com

Github link:github.com/AshutoshGupta221299/osproject

Code:question 2

Problem explaination:

We have to design a scheduler which schedules the process as concidering arrival time and burst time requirement of the process , the scheduler schedules the process by interrupting the processor after every 6 units time and does concider 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.

Considering the following units for reference.

Process    Arrival time    Burst time

P1    0     20

P2    5     36

P3     13     19

P4     26     42

We have to 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.

Algorithm:

1. For First and Second Iteration.

2. For first iteration, timeQuantum =6 and for second Iteration timeQuantum =10.

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

If(arrivalTime < TimeDuration):

If(burstTime > timeQuantam):{

Decrease burstTime by timeQuantum;

}

Else:{

TimeDuration += burstTime;

CompilationTime = TimeDuration;

TurnAroundTime = CompilationTime - arrivalTime;

burstTime -= timeQuantum;

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

}

4. then we Sort remaining processes according to remaining burstTime.

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

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

TimeDuration += burstTime;

CompilationTime=timeDuration;

TurnAroundTime = CompilationTime - arrivalTime;

}

7. we Print all the process. And Average TurnAroundTime 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 remainingtime[10],int currenttime,int arrivaltime[10], int bursttime[10]);

main()

{

int processnumber,j,no,currtime,remainingprocess,indicator,time\_quan,wait,tut,arrivaltime[10],bursttime[10],remainingtime[10],x=1;

indicator = 0;

wait = 0;

tut = 0;

printf("Enter number of processes ");

scanf("%d",&no);

remainingprocess = no;

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

for(processnumber = 0;processnumber < no;processnumber++)

{

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

printf("Arrival time = ");

scanf("%d",&arrivaltime[processnumber]);

printf("Burst time = ");

scanf("%d",&bursttime[processnumber]);

remainingtime[processnumber]=bursttime[processnumber];

}

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

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

time\_quan=6;

currtime=0;

for(processnumber=0;remainingprocess!=0;)

{

if(remainingtime[processnumber]<=time\_quan && remainingtime[processnumber]>0)

{

currtime+=remainingtime[processnumber];

remainingtime[processnumber]=0;

indicator=1;

}

else if(remainingtime[processnumber]>0)

{

remainingtime[processnumber]-=time\_quan;

currtime+=time\_quan;

}

if(remainingtime[processnumber]==0 && indicator==1)

{ printf("%d",processnumber);

remainingprocess--;

printf("P %d",processnumber+1);

printf("\t\t\t%d",currtime-arrivaltime[processnumber]);

printf("\t\t\t%d\n",currtime-bursttime[processnumber]-arrivaltime[processnumber]);

wait+=currtime-arrivaltime[processnumber]-bursttime[processnumber];

tut+=currtime-arrivaltime[processnumber];

indicator=0;

}

if(processnumber==no-1){

x++;

if(x==2){

processnumber=0;

time\_quan=10;

printf("The time quantum for second round is 10. \n");

}

else{

break;

}

}

else if(currtime >= arrivaltime[processnumber+1]){

processnumber++;

}

else{

processnumber=0;

}

}

rr(no,remainingtime,currtime,arrivaltime,bursttime);

return 0;

}

void rr(int no,int remainingtime[10],int currenttime,int arrivaltime[10], int bursttime[10]){

float avg\_wait,avg\_tut;

int i,j,n=no,temp,btime[20],processnumber[20],w\_time[20],turnaroundtime[20],total=0,loc;

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

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

{

btime[i]=remainingtime[i];

w\_time[i]=currenttime-arrivaltime[i]-btime[i];

processnumber[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=processnumber[i];

processnumber[i]=processnumber[loc];

processnumber[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++)

{

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

total=total + turnaroundtime[i];

printf("\nP%d\t\t\t%d\t\t\t%d\t\t\t%d",processnumber[i],btime[i],w\_time[i],turnaroundtime[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:

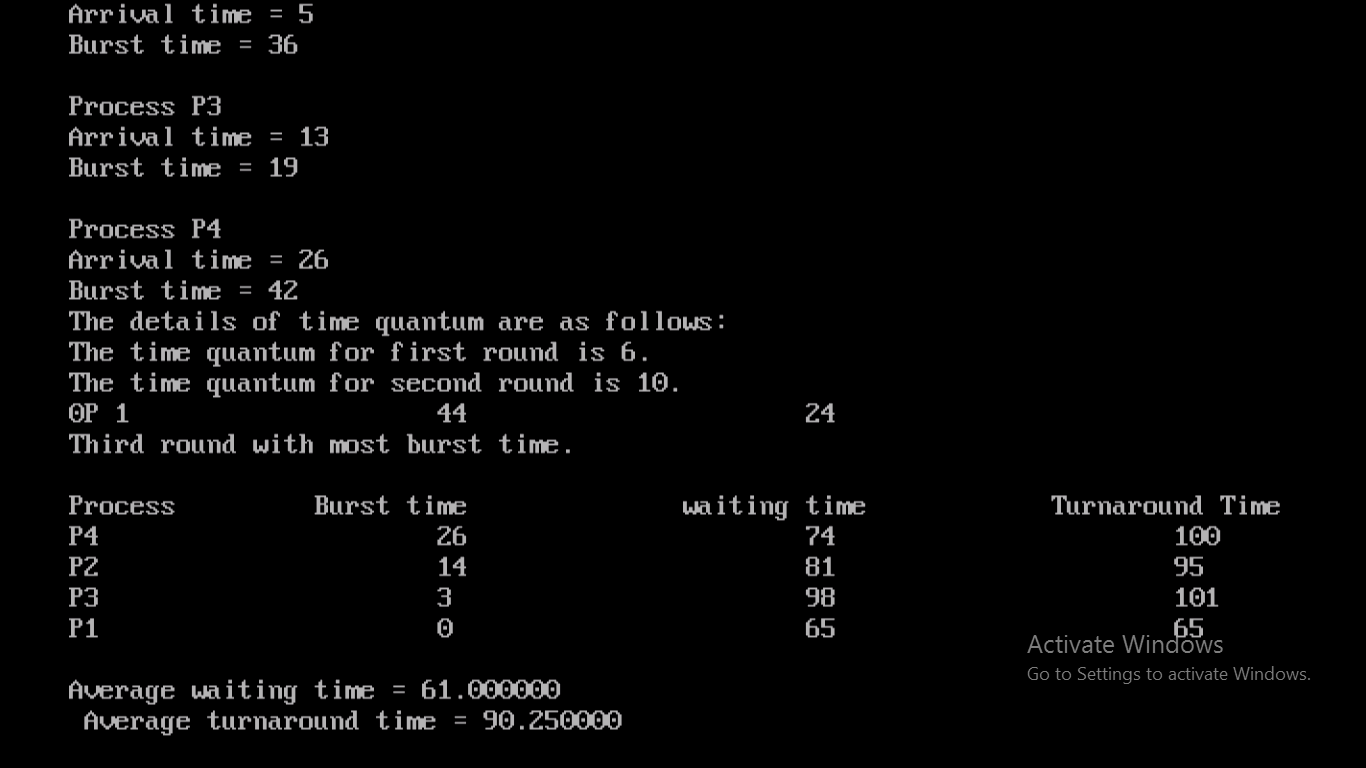
Number of processes cannot be a negative number, it has to be a positive number

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 61 and 90.25 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 6 and then second iteration with time quantum 10 and displayed the stats after 2 iterations in which p1 was complete and we have done third iteration with most 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 6 and then second iteration with time quantum 10 and displayed the stats after 2 iterations in which p1 was complete and we have done third iteration with most burst time and solved accordingly and again displayed the output and average waiting time and average turn around time.