Student Name:ANIKET RAJ
Reg_no:11811888
Roll_no:03
Section:K18FR
Email Address:

aniketraj771@gmail.com

GitHub Link:

https://github.com/Aniket-rawat/operatingggsyss.git

School of Computer Science & Engineering Lovely professional University Phagwara, Punjab.

INDEX

QUESTION;1

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

CODE:

```
//Operating system Schedular code.


#include<stdio.h> /

#include<conio.h>


void rr(int no,int remt[10],int Cur_t,int arT[10], int bsT[10]);//


main()

{

        0

    int Proc_no,j,no,CurT,RemProc,indicator,time_quan,wait,tut,arT[10],bsT[10],remt[10],x=1;

        indicator = 0; //it assign  when process is fully terminated
        wait = 0;

        tut = 0;  //tut=turn around time

        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]); //art = arrival time

                printf("Burst time = ");   //when process enter into ready queue

                scanf("%d",&bsT[Proc_no]);

                remt[Proc_no]=bsT[Proc_no];   //bsT is reffering to burst time

        }

        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;)          //checks if remaining process !0

        {


                if(remt[Proc_no]<=time_quan && remt[Proc_no]>0)   //here it checks that
    remaining burst

                                            // time is less or equal to time quantum

                {

                        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)
                                    //indicator flag as 1 as soon process is terminated
    { 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]);        //remaining
burst time

            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");      //in
second iteration the time quantumis 6
                        }
                        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;    //avg_wait :avg waiting time
  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];
                                    //hera the time slice or time quantum
                    Proc_no[i]=i+1;           //is takes as the unit in which remaining process is
terminated
    }

    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];   //turn around time =burst time + waiting time
        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;     //average turn aound time =total tunaround time /no of process

    printf("\n\nAverage waiting time = %f",avg_wait);
    printf("\n Average turnaround time = %f\n",avg_tut);

}
```

DESCRIPTION:

The complexity of the following code is n^2;

The code can be divided into 3 main parts which are 2 different functions and the code inside the main function.

The purpose of the function Firstact is to decide which process will go first and which will remain in the waiting state.

The purpose of the implement function is to print which procees is being implemented and which functions are in the waiting state. It also appends the active processes to a list which are later used in the main function.

The 2 for loop division part of the code is used for the calculation of the average time and average turn around time of the processes.

Limitation:

The code has give wrong input if the arrival time of the processes is negative hence all the values should be checked in input.

Bibiliography:

"Operating System Concepts" by Silberschatz, Galvin, and Gagne :

https://www.youtube.com/watch?v=TxjIlNYRZ5M