LOVELY PROFESSIONAL UNIVERSITY.

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Student Name: Venkata Rohith Raghupatruni.

Student ID: 11808736

Roll no: 68

Email Address: rohithprince2000@gmail.com.

GitHub Link: https://github.com/rohithprince/process_scheduling.

Code:

Q5. Sudesh Sharma is a Linux expert who wants to have an online system where he can handle student queries. Since there can be multiple requests at any time he wishes to dedicate a fixed amount of time to every request so that everyone gets a fair share of his time. He will log into the system from 10am to 12am only. He wants to have separate requests queues for students and faculty. Implement a strategy for the same. The summary at the end of the session should include the total time he spent on handling queries and average query time

```
#include<stdio.h>

#include<string.h>

struct schedule{

        int Job_id;

        int Arri_time;

        int Bur_time;

        int Complet_time;

        int Remain_time;//rbt

}fqueue[100],squeue[100],rqueue[100];

int z, zfc=0,zsc=0, readycount=0;

int timequantum;


void Schedule_by_round_robin(){

        int time= rqueue[0].Arri_time, cont=0, pp=0, i, prc;

        while(time!=120 && pp!=readycount){
                for(i=0; i<=cont; i++){
                        if(rqueue[i].Remain_time > timequantum){
                                time=time + timequantum;
                                rqueue[i].Remain_time -= timequantum;
                        }
                        else if(rqueue[i].Remain_time <=timequantum &&
rqueue[i].Remain_time !=0){
                                time += rqueue[i].Remain_time;
                                rqueue[i].Remain_time =0;
                                rqueue[i].Complet_time = time;
                                pp++;
                        }
                        else;
                }
                int first = cont+1;
                for(prc= first; prc<readycount; prc++){
                        if(rqueue[prc].Arri_time <= time){
                                cont++;
                        }
                }
        }
}


////
```

```
///
void Taking_input_both_queues(){
        int aic=0, bic= 0, Minimum, Flag;
        if( zfc!=0  && zsc!=0){
                while(aic<zsc && bic<zfc){
                        if(fqueue[bic].Arri_time == squeue[aic].Arri_time){
                                rqueue[readycount] = fqueue[bic];
                                readycount++;
                                bic++;
                                rqueue[readycount]= squeue[aic];
                                readycount++;
                                aic++;
                        }
                        else if(fqueue[bic].Arri_time < squeue[aic].Arri_time){
                                rqueue[readycount]= fqueue[bic];
                                readycount++;
                                bic++;
                        }
                        else if(fqueue[bic].Arri_time > squeue[aic].Arri_time){
                                rqueue[readycount]= squeue[aic];
                                readycount++;
                                aic++;
                        }
                        else;
                }
        if(readycount != (zfc+zsc)){
            if(zfc!=bic){
                while(bic!=zfc){
                    rqueue[readycount]= fqueue[bic];
                    readycount++;
                    bic++;
                }
            }
            else if(zsc!=aic){
                while(aic!=zsc){
                    rqueue[readycount]= squeue[aic];
                    readycount++;
                    aic++;
                }
            }
        }
    }
    else if(zfc==0){
        while(aic!=zsc){
            rqueue[readycount]= squeue[aic];
            readycount++;
            aic++;
        }
    }
    else if(zsc==0){
```

```
                while(bic!=zfc){
                rqueue[readycount]= fqueue[bic];
                        readycount++;



                        bic++;
                }
        }
        else {
                printf("\n No valid Jobs available\n");
        }}
void Data(){

        int Queryt,i, k;
        printf("Enter total no of queries: "); scanf("%d", &z);
        if(z==0) { printf("\n No queries has be accepeted\n"); }
        else{
                printf("\nEnter timequantum for each Process: "); scanf("%d", &timequantum);

                for(i=0; i<z; i++){
                        printf("n:Press 1(faculty) or 0(student) "); scanf("%d",&Queryt);
                        if(Queryt==1){
                                printf("faculty Query Id: "); scanf("%d", &fqueue[zfc].Job_id);
                                printf("Arrival Time: "); scanf("%d", &k);
                                if(k<1000 || k>1200){
```

```
                        printf("\n Entered wrong slot time ");
                        Data();




                    }
                    else{fqueue[zfc].Arri_time= k-1000;}
                    printf("Burst time: "); scanf("%d", &fqueue[zfc].Bur_time);



fqueue[zfc].Remain_time= fqueue[zfc].Bur_time;
                    zfc++;
                } else{
                    printf("student Query Id: "); scanf("%d", &squeue[zsc].Job_id);
                    printf("Arrival Time: "); scanf("%d", &k);
                    if(k<1000 || k>1200){
                        printf("\nEntered wrong slot time\n");
                        Data();
                    }
                    else {squeue[zsc].Arri_time= k-1000; }
                    printf("Burst time: "); scanf("%d", &squeue[zsc].Bur_time);
squeue[zsc].Remain_time= squeue[zsc].Bur_time;
                    zsc++;
                }
            }
```

```
    }}

void Display_output(){

    int p=0, tot=0, sum=0;
    double Avg;
    printf("\nHere is the summary....\n");
    printf("\nQuery ID\tArrival Time\tRessolving Time\tCompletion Time\tTurn Around Time\tWaiting Time");
    for(p; p<readycount; p++){
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t\t%d",
        rqueue[p].Job_id, (rqueue[p].Arri_time+1000), rqueue[p].Bur_time, (rqueue[p].Complet_time+1000), (rqueue[p].Complet_time-rqueue[p].Arri_time), ((rqueue[p].Complet_time-rqueue[p].Arri_time)- rqueue[p].Bur_time));
        tot= rqueue[p].Complet_time;
        sum+= (rqueue[p].Complet_time-rqueue[p].Arri_time);}
            Avg = sum/readycount;
    printf("\n\nTotal time Spent for all queries: %d", tot);
    printf("\nAverage query time taken is : %f", Avg);//lf
    printf("\nAll queries have been solved");}
```

```
void main(){




            printf("\n \t \t \t \t \t------Important Notice------\n");

            printf("**Available from 10AM to 12PM only *.");

            printf("\n \n** Please enter arrival time of process in format 1000 to 1200.\n for example
10AM=1000,10:30AM=1030.*");

            printf("\n \n** Time units are in minutes.\n\n\n");

            Data();

            Taking_input_both_queues();

            Schedule_by_round_robin();

            Display_output();

}
```

DESCRIPTION:

The give problem is based upon process scheduling of faculty and students, where they could enter the arrival time and burst time of their query. These queries can be compared to different processes in an operating system where each process needs resources and time for its execution. These were handled by the CPU.In this problem sudesh sharma wants to dedicate fixed amount of time for each query of faculty and students, we use Round Robin method to solve this problem, by dividing the requests into two seperate categories ie,faculty queries and student queries. Then we solve the problem by allocating them required resources based upon their priorities.

ALGORITHM :

1.Take process arrival time,burst time and quantum time from the user. Complexity: O(n)

2.The Arrival time is sorted in ascending order. O(n^2)

3.The processes will be sorted in order of their priority O(nlogn)

4.According to the time quantum ,the processes will execute for given time and then go into the waiting state.

5.Other process in the ready state will be executed having the highest priority.

6.Repeat step 4 and 5until all processes are done.

7.Find waiting time,completion time and turn around time for all processes. O(n)

Turn Around Time : rqueue[p].Complet_time-rqueue[p].Arri_time

Waiting Time : rqueue[p].Complet_time-rqueue[p].Arri_time)- rqueue[p].Bur_time

8.Find Average Waiting TimeO(n)

Average Turn Around TimeComplexity: O(n)

CONSTRAINTS:

1.The entered time must be like 1000 for 10:AM,1030 for 10:30 AM.

2.take the input in sorted order.

3.Each process has to be executed for a specified time period and then it will be stored in waiting state.

4. The time quantum must be given by the user(compulsory).

BOUNDARY CONDITIONS:

1.The arrival range must be 1000 to 1200 only.

2.the process must be in ascending order.

3.They can different time interevals.

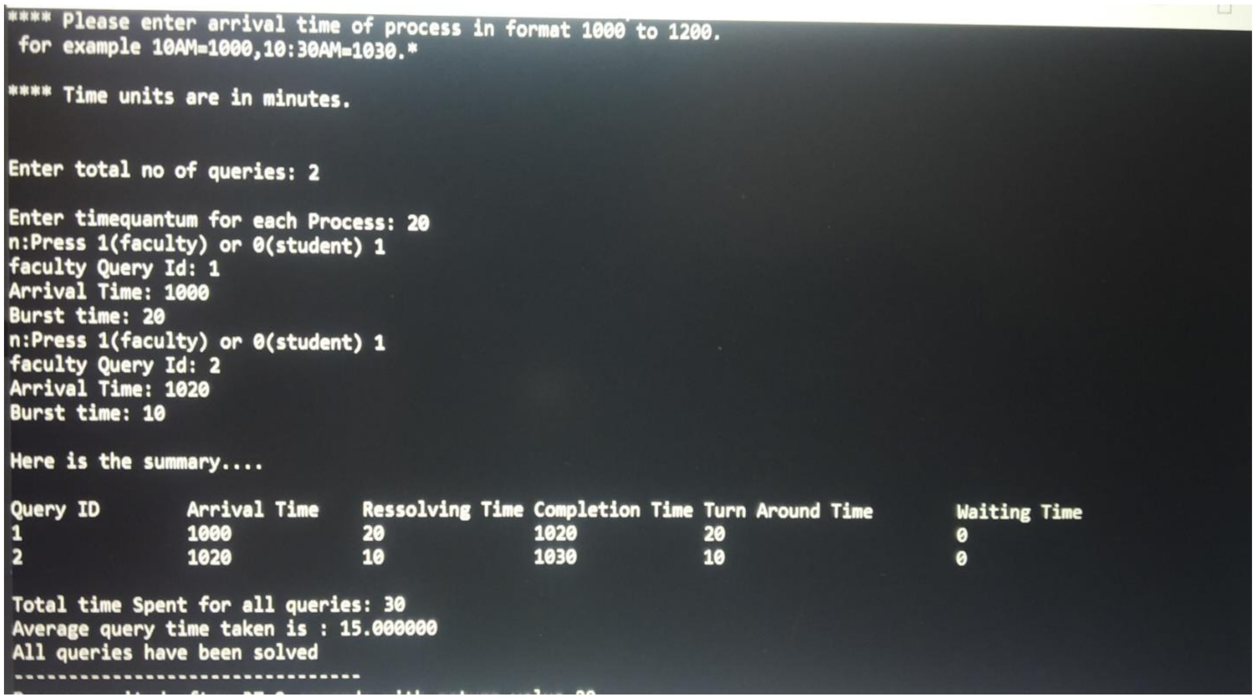
TEST CASES:

CASE-1: For handling 2 faculty queries.

Process	**AT**	**BT**
Faculty 1	1000	20
Faculty 2	1020	10

Timequantum : 20.

Screen shot:



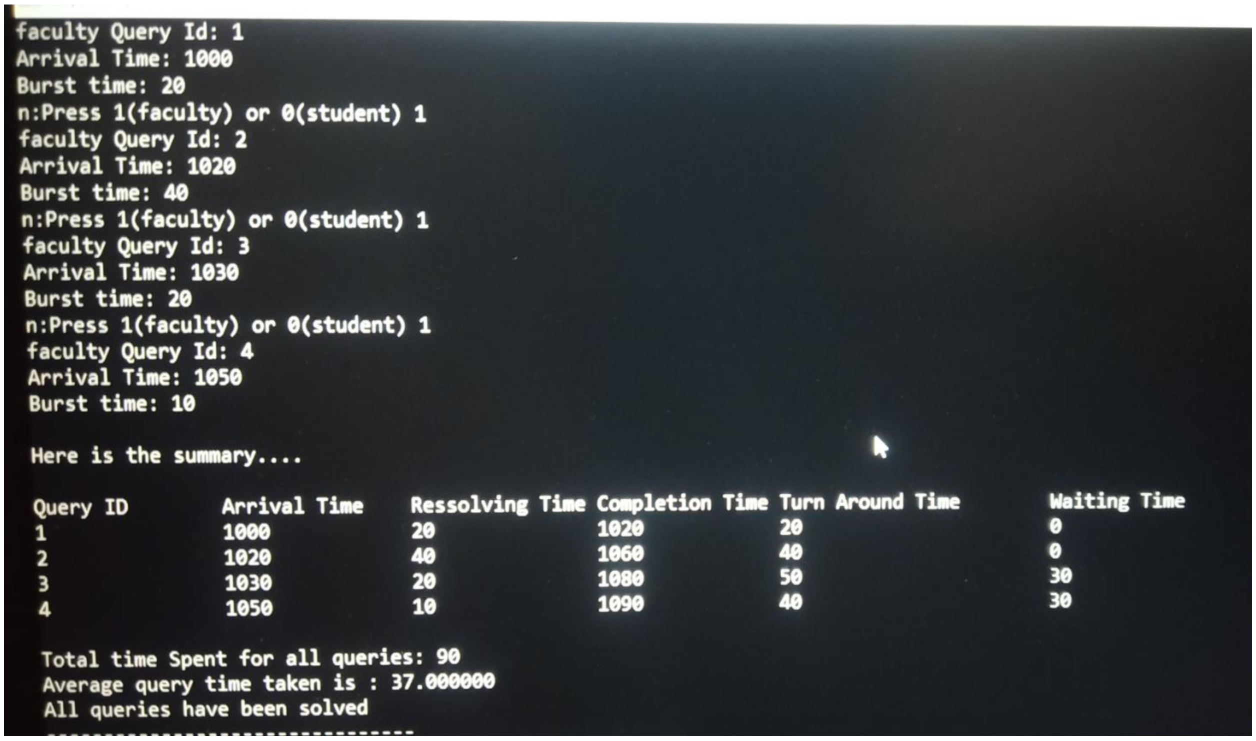
```
**** Please enter arrival time of process in format 1000 to 1200.
 for example 10AM=1000,10:30AM=1030.*

**** Time units are in minutes.


Enter total no of queries: 2

Enter timequantum for each Process: 20
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 1
Arrival Time: 1000
Burst time: 20
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 2
Arrival Time: 1020
Burst time: 10

Here is the summary....

Query ID       Arrival Time    Ressolving Time Completion Time Turn Around Time       Waiting Time
1              1000            20              1020            20                     0
2              1020            10              1030            10                     0

Total time Spent for all queries: 30
Average query time taken is : 15.000000
All queries have been solved
--------------------------------
```

CASE-2: For handling 4 faculty queries.

Process	AT	BT
Faculty 1	1000	20
Faculty 2	1020	40
Faculty 3	1030	20
Faculty 4	1050	10

Timequantum : 20.

Screen shot:



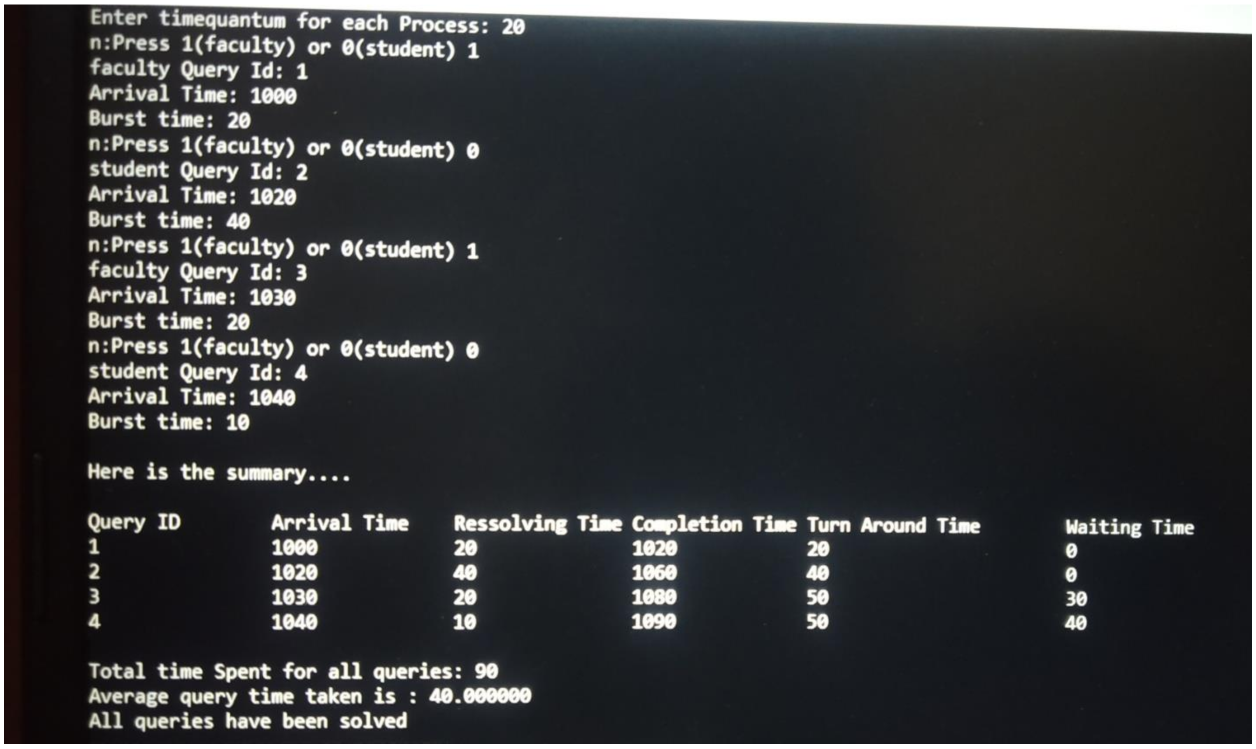
```
faculty Query Id: 1
Arrival Time: 1000
Burst time: 20
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 2
Arrival Time: 1020
Burst time: 40
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 3
Arrival Time: 1030
Burst time: 20
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 4
Arrival Time: 1050
Burst time: 10

Here is the summary....

Query ID        Arrival Time    Ressolving Time Completion Time Turn Around Time        Waiting Time
1               1000            20              1020            20                      0
2               1020            40              1060            40                      0
3               1030            20              1080            50                      30
4               1050            10              1090            40                      30

Total time Spent for all queries: 90
Average query time taken is : 37.000000
All queries have been solved
```

CASE-3: For handling 4 mixed queries queries.

Process	**AT**	**BT**
Faculty 1	1000	20
Faculty 2	1020	40
Faculty 3	1030	20
Faculty 4	1050	10

Timequantum : 20.

Screen shot:



```
Enter timequantum for each Process: 20
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 1
Arrival Time: 1000
Burst time: 20
n:Press 1(faculty) or 0(student) 0
student Query Id: 2
Arrival Time: 1020
Burst time: 40
n:Press 1(faculty) or 0(student) 1
faculty Query Id: 3
Arrival Time: 1030
Burst time: 20
n:Press 1(faculty) or 0(student) 0
student Query Id: 4
Arrival Time: 1040
Burst time: 10

Here is the summary....

Query ID        Arrival Time    Ressolving Time Completion Time Turn Around Time        Waiting Time
1               1000            20              1020            20                      0
2               1020            40              1060            40                      0
3               1030            20              1080            50                      30
4               1040            10              1090            50                      40

Total time Spent for all queries: 90
Average query time taken is : 40.000000
All queries have been solved
```

GITHUB LINK: https://github.com/rohithprince/process_scheduling.