OS Simulation Based Assignment Assessment Rubric

Student Name: - Prakash Singh

Roll No.-12

Section-K18MS

Registration no.-11803164

Email Address:123pksingh786@gmail.com

GitHub Link: **https://github.com/impksingh/os-project.git**

Description:

Problem:12

Three students (a, b, c) are arriving in the mess at the same time. The id numbers of these students are 2132, 2102, 2453 and the food taken time from the mess table is 2, 4 and 8 minutes. If the two students have same remaining time so it is broken by giving priority to the students with the lowest id number. Consider the longest remaining time first (LRTF) scheduling algorithm and calculate the average turnaround time and waiting time.

For performing LRTF we follow 3 steps:-

- First, sort the processes in increasing order of their Arrival Time.,if it is given otherwise first come first serve apply.
- Choose the process having least arrival time but with most Burst Time. Then process it for 1 unit. Check if any other process arrives upto that time of execution or not.
- Repeat the above both steps until execute all the processes.

ALGORITHM:-

Theoretical Explanation:-

1. LRTF is to be used. Student C has the longest remaining time. So the serving starts with C.
2. After C is served for 4 minutes, the remaining time of B and C are equal. B has the lowest ID number. So the serve moves onto B.
3. After B is served for 2 minutes, the reaming time of A and B are equal. But still, B has the lowest ID number. So B is served.

4. Since his “food taken time” is 4 minutes, he is done with the job.

5. Comparing the remaining students A and C, C has a longer waiting time i.e., 4 min (he was already served 4 minutes, total time being 8 min). So, C is continued with the service.

After serving him for 2 minutes, the remaining time of C and A are equal. A has the lowest ID number. So the serve moves onto A.

After serving A for 2 minutes, he is done with the job since his “food taken time” is 2 minutes.

The serve moves onto C again to process the remaining 2 min of his “food taken time”.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

c c c c b b b b c c a a c c

Turn Around Time(TAT) = Completion Time(CT) - Arrival Time(AT)

Wait Time (WT) = Turn Around Time(TAT) - Burst Time(BT)

A completes the job in 12 min, arrival time is 0.

B completes the job in 8 min, arrival time is 0.

C completes the job in 14 min, arrival time is 0.

Turn Around Time(TAT) = Completion Time(CT) - Arrival Time(AT)

TAT(A) = 12 - 0 = 12 min

TAT(B) = 8 - 0 = 8 min

TAT(C) = 14 - 0 = 14 min

Given;

A’s Burst Time(BT) = 2 min

B’s Burst Time(BT) = 4 min

C’s Burst Time(BT) = 8 min

Here, Burst Time(BT) = Food Taken Time

Wait Time (WT) = Turn Around Time(TAT) - Burst Time(BT)

WT(A) = 12 - 2 = 10

WT(B) = 8 - 4 = 4

WT(C) = 14 - 8 = 6

CODE:-

```
#include <stdio.h>
struct student
{
   int STD_ID,Time_for_food,WT,TAT;

};

void get_data(struct student list[], int s);
void show(struct student list[], int s);
void scheduling(struct student list[], int s);
void WT(struct student list[], int n);
void TAT(struct student list[], int n);

int main()
{
   struct student data[20];
   int n,i;
   char c='n';
   do
         {
   printf("NUMBER OF STUDENT WANTS TO EAT IN MESS :- ");
   scanf("%d", &n);
   get_data(data, n);
   scheduling(data, n);
   WT(data,n);
   TAT(data,n);
   show(data, n);
   printf("=======================================WANT TO TRY FOR MORE
VALUES================================== ");
   scanf("%s",&c);
   }while(c=='y');
   return 0;
}

void get_data(struct student list[80], int s)
{
   int i;
   for (i = 0; i < s; i++)
   {

      printf("\nEnter Student id %d\t", i + 1);
      scanf("%d", &list[i].STD_ID);

      printf("Enter time taken for food (minuts) for Student %d\t", i + 1);
```

```
        scanf("%d", &list[i].Time_for_food);
    }
}
void show(struct student list[80], int s)
{
    int i,AvgWT=0,AvgTAT=0;
          int TotalWatingTime=0,TotalTAT=0;
    printf("\n\nOutput according to LRTF\n");
    printf("\n|Student id\tTime_for_food\tWT\t\tTAT  |");


    for (i = 0; i < s; i++)
    {
        printf("\n|%d\t\t%d\t\t%d\t\t%d\t\t|", list[i].STD_ID,
list[i].Time_for_food,list[i].WT,list[i].TAT);
                    TotalWatingTime= TotalWatingTime+list[i].WT;
                    TotalTAT= TotalTAT+list[i].TAT;
          }
          printf("\n\nTotal Waiting Time is: = %d",TotalWatingTime);
          printf("\nTotal Turn around Time is: = %d\n\n",TotalTAT);
          printf("\n\ntAverage Waiting Time is: = %d",TotalWatingTime/s);
          printf("\nAverage Turn around Time is: = %d\n\n",TotalTAT/s);
}

void scheduling(struct student list[80], int s)
{
    int i, j;
    struct student temp;

    for (i = 0; i < s - 1; i++)
    {
        for (j = 0; j < (s - 1-i); j++)
        {
            if (list[j].Time_for_food < list[j + 1].Time_for_food)
            {
                temp = list[j];
                list[j] = list[j + 1];
                list[j + 1] = temp;
            }
            else if(list[j].Time_for_food == list[j + 1].Time_for_food)
            {
                    if(list[j].STD_ID > list[j + 1].STD_ID)
                    {
                    temp = list[j];
                list[j] = list[j + 1];
                list[j + 1] = temp;
                }
                        }
        }
    }
```

```
}


void WT(struct student list[80], int n)
{
int j,total;
list[0].WT=0;
   for(j=1;j<n;j++)
   {
      list[j].WT=list[j-1].WT+list[j-1].Time_for_food;
   }
}


void TAT(struct student list[80], int n)
{
int j,total;

         for(j=0;j<n;j++)
        {
        list[j].TAT=list[j].WT+list[j].Time_for_food;
        }
}
```

Advantages:-

SRTF algorithm makes the processing of the jobs faster than SJN algorithm, given it's overhead charges are not counted.

Disadvantages:-

The context switch is done a lot more times in SRTF than in SJN, and consumes CPU's valuable time for processing. This adds up to it's processing time and diminishes it's advantage of fast processing.

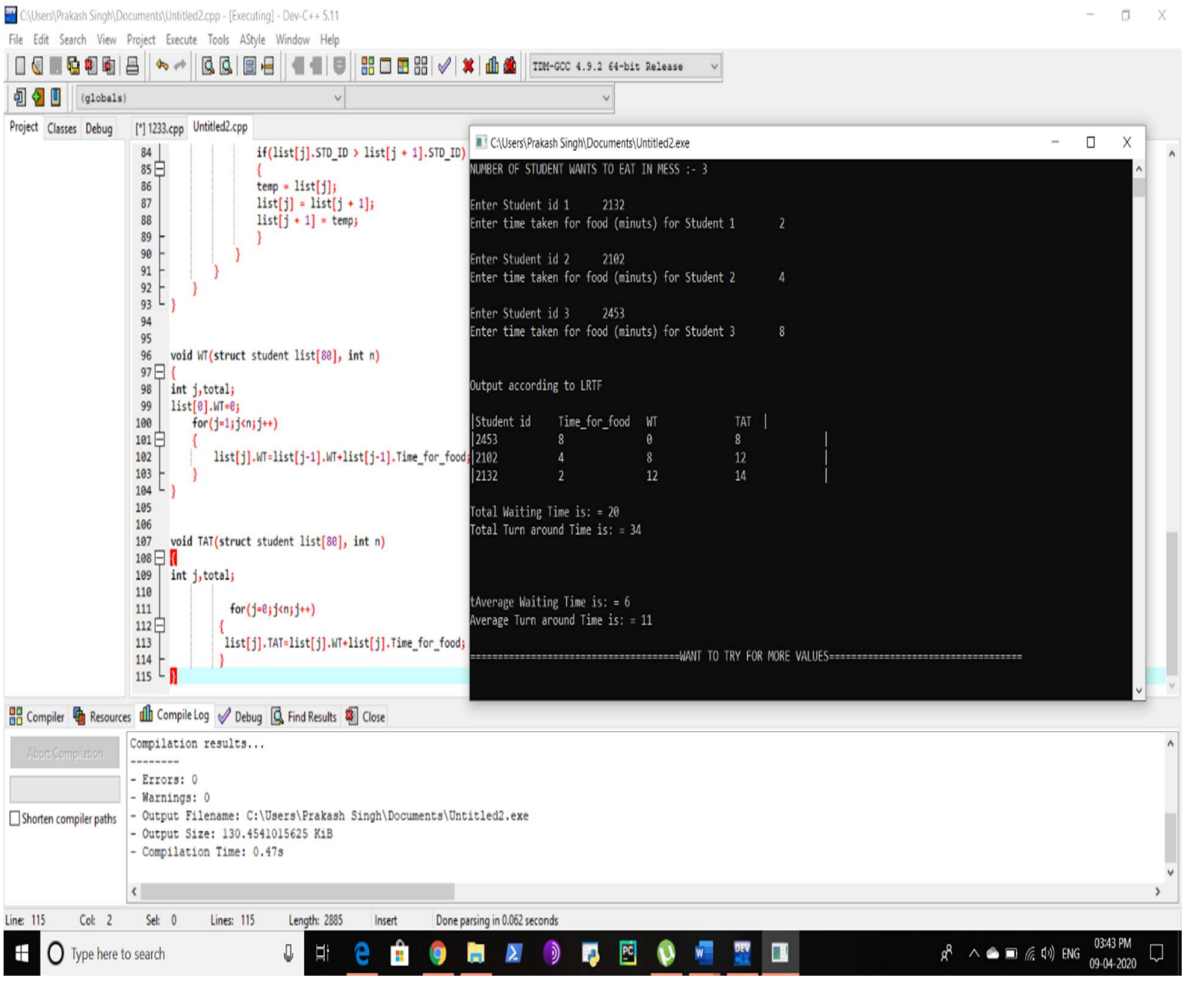
CONSTRAINTS:-

Like shortest job next scheduling, shortest remaining time scheduling is rarely used outside of specialized environments because it requires accurate estimates of the runtime of each process.

TEST RESULTS:-

Output is obtained as-

- **The required Average Turn Around Time** = (12+8+14)/3 = 11.33 minutes
- **The required Average Wait Time** = (10+4+6)/3 = 6.67 minutes



GITHUB LINK-https://github.com/impksingh/os-project.git