**Student Name:** Anukul Singh

**Student ID:** 11805001 (37) K18JF

**Email Address:** [anukulclasher@gmail.com](mailto:anukulclasher@gmail.com)

**GitHub Link:**

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.

**Ans:**

LRTF is to be used. Student C has the longest remaining time. So the serving starts with C.

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.

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.

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

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

**Therefore,**

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

.

Algorithm:

* **Step-1**: Create a structure of process containing all necessary fields like AT (Arrival Time), BT (Burst Time), CT (Completion Time), TAT (Turn Around Time), WT (Waiting Time).
* **Step-2**: Sort according to the AT;
* **Step-3**: Find the process having Largest Burst Time and execute for each single unit. Increase the total time by 1 and reduce the Burst Time of that process with 1.
* **Step-4**: When any process has 0 BT left, then update the CT (Completion Time of that process CT will be Total Time at that time).
* **Step-5**: After calculating the CT for each process, find TAT and WT.

**Purpose**

This is a pre-emptive version of Longest Job First (LJF) scheduling algorithm. In this scheduling algorithm, we find the process with maximum remaining time and then process it. We check for the maximum remaining time after some interval of time (say 1 unit each) to check if another process having more Burst Time arrived up to that time.

Description:

There are 3 students to be served. Since they arrive at the same, we need an algorithm as to how the students are to be served food.

Given that;

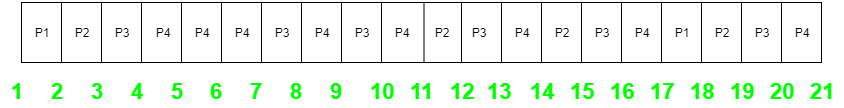
LRTF (Longest Remaining Time Algorithm) is to be used.

Also, the questions states that, if two students have the same remaining time, the student with the lowest id number is served first.

**Procedure**

* **Step-1**: First, sort the processes in increasing order of their Arrival Time.
* **Step-2**: Choose the process having least arrival time but with most Burst Time. Then process it for 1 unit. Check if any other process arrives up to that time of execution or not.
* **Step-3**: Repeat the above both steps until execute all the processes. Example: Consider the following table of arrival time and burst time for four processes P1, P2, P3 and P4.

| **Process** | **Arrival time** | **Burst Time** |
| --- | --- | --- |
| P1 | 1 ms | 2 ms |
| P2 | 2 ms | 4 ms |
| p3 | 3 ms | 6 ms |
| p4 | 4 ms | 8 ms |

[](https://github.com/mrjatinchauhan/LRTF-Scheduling/blob/master/img/GANT.png)**Gantt chart:**

Code:

**General Terms:**

* **Arrival Time**: Time at which the process arrives in the ready queue.
* **Completion Time**: Time at which process completes its execution.
* **Burst Time**: Time required by a process for CPU execution.
* **Turn Around Time**: Time Difference between completion time and arrival time.
* **Waiting Time**: Time Difference between turnaround time and burst time.

Code:

//Project done by Anukul Singh

#include <stdio.h>

struct student

{

int StudentId;

int FoodTakenTime;

int WaitingTime;

int TurnAroundTime;

};

void accept(struct student list[], int s);

void display(struct student list[], int s);

void scheduling(struct student list[], int s);

void waitingTime(struct student list[], int n);

void turnAroundTime(struct student list[], int n);

int main()

{

struct student data[20];

int n,i;

char c='n';

do

{

printf("Please enter the No. of Students wants to eat in mess? : ");

scanf("%d", &n);

accept(data, n);

scheduling(data, n);

waitingTime(data,n);

turnAroundTime(data,n);

display(data, n);

printf("Want to continue? press 'y' : ");

scanf("%s",&c);

}while(c=='y');

return 0;

}

void accept(struct student list[80], int s)

{

int i;

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

{

printf("\n\nEnter data for Student #%d", i + 1);

printf("\nEnter Student id : ");

scanf("%d", &list[i].StudentId);

printf("Enter time taken for food (minuts): ");

scanf("%d", &list[i].FoodTakenTime);

}

}

void display(struct student list[80], int s)

{

int i,AvgWaitingTime=0,AvgTurnAroundTime=0;

int TotalWatingTime=0,TotalTurnAroundTime=0;

printf("\n\n\t\t\tOutput according to LRTF\n");

printf("\n\t\t\t|===============================================================|");

printf("\n\t\t\t|Student id\tFoodTakenTime\tWaitingTime\tTurnAroundTime |");

printf("\n\t\t\t|===============================================================|");

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

{

printf("\n\t\t\t|%d\t\t%d\t\t%d\t\t%d\t\t|", list[i].StudentId, list[i].FoodTakenTime,list[i].WaitingTime,list[i].TurnAroundTime);

printf("\a\n\t\t\t|---------------------------------------------------------------|");

TotalWatingTime= TotalWatingTime+list[i].WaitingTime;

TotalTurnAroundTime= TotalTurnAroundTime+list[i].TurnAroundTime;

}

printf("\n\n\t\t\tTotal Waiting Time is: = %d",TotalWatingTime);

printf("\n\t\t\tTotal Turn around Time is: = %d\n\n",TotalTurnAroundTime);

printf("\n\n\t\t\tAverage Waiting Time is: = %d",TotalWatingTime/s);

printf("\n\t\t\tAverage Turn around Time is: = %d\n\n",TotalTurnAroundTime/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].FoodTakenTime < list[j + 1].FoodTakenTime)

{

temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

}

else if(list[j].FoodTakenTime == list[j + 1].FoodTakenTime)

{

if(list[j].StudentId > list[j + 1].StudentId)

{

temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

}

}

}

}

}

void waiting Time (struct student list[80], int n)

{

int j,total;

list[0].WaitingTime=0;

for(j=1;j<n;j++)

{

list[j].WaitingTime=list[j-1].WaitingTime+list[j-1].FoodTakenTime;

}

}

void turnAroundTime (struct student list [80], int n)

{

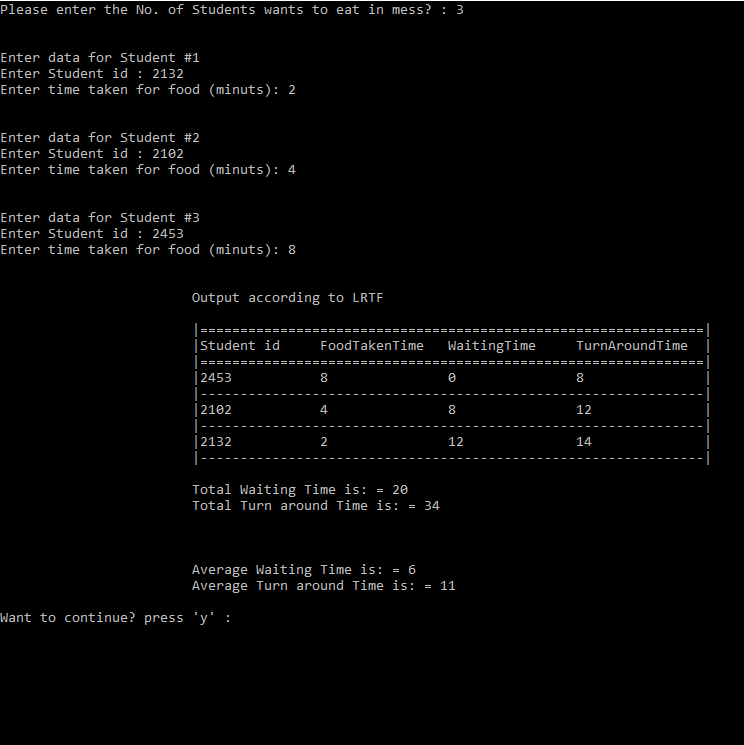
int j,total;

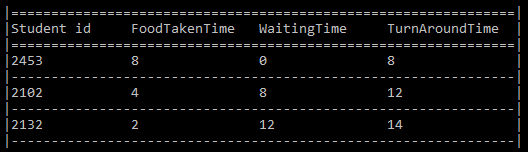
for(j=0;j<n;j++)

{list[j]. TurnAroundTime=list[j].WaitingTime+list[j].FoodTakenTime;

}

}

[](https://github.com/mrjatinchauhan/LRTF-Scheduling/blob/master/img/ScreenshotSnippets.png)**Output:**

[](https://github.com/mrjatinchauhan/LRTF-Scheduling/blob/master/img/ScreenshotTable.png)

**GitHub Link**

[**https://github.com/AnukulClasher/CSE-316-**](https://github.com/AnukulClasher/CSE-316-)