Project Topic :-

Faculty Query- Time Scheduler

[USing RR Algorithm]

Submitted for:-

Operating System (UCS303)

Submitted To:-

Dr. Shashank S. Singh Sir

Submitted By:-

Aryan Varshney – 101903605

Rohan Kumar -101903625

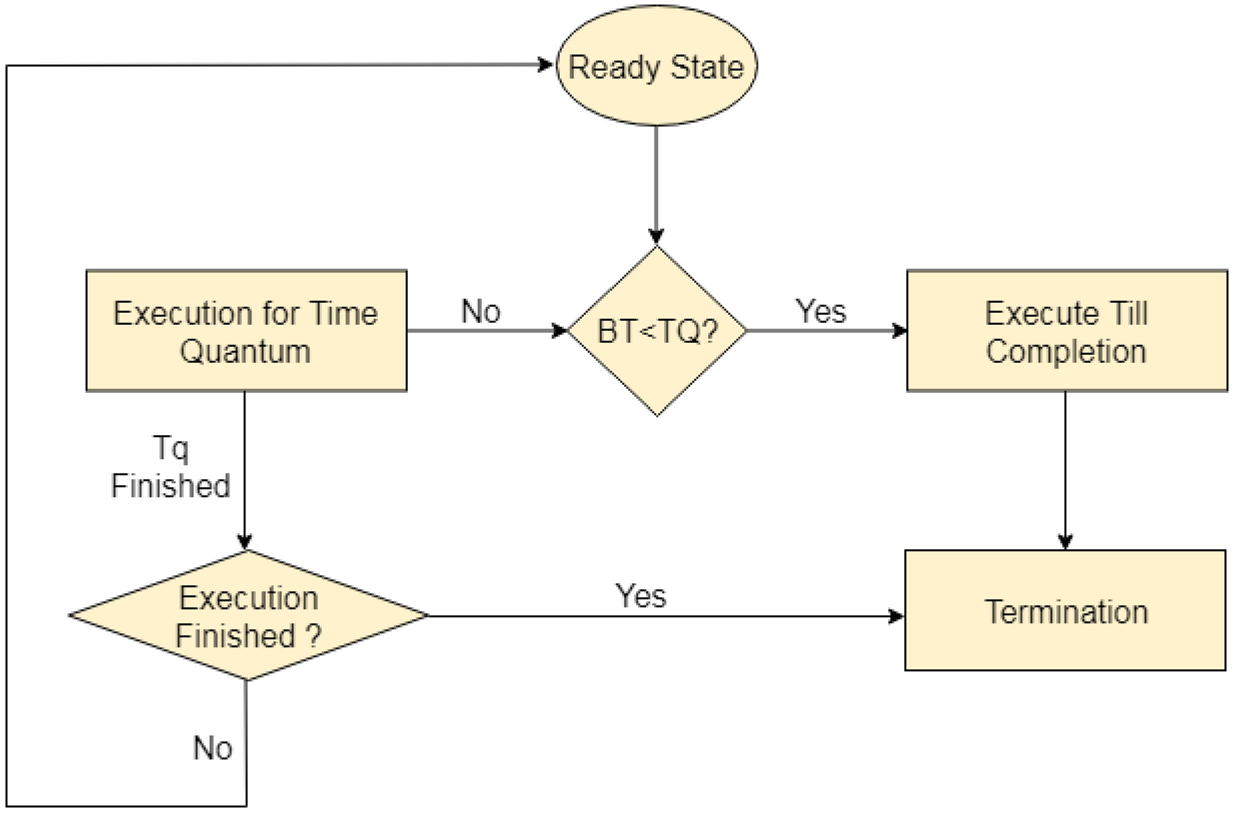
THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY,

(Deemed To Be University), Patiala, Punjab, India

<u>Introduction</u> :-

Round Robin scheduling algorithm is one of the most popular scheduling algorithm which can actually be implemented in most of the operating systems. This is the **pre-emptive version** of first come first serve scheduling. The Algorithm focuses on Time Sharing. In this algorithm, every process gets executed in a **cyclic way**. A certain time slice is defined in the system which is called time **quantum**. Each process present in the ready queue is assigned the CPU for that time quantum, if the execution of the process is completed during that time then the process will **terminate** else the process will go back to the **ready queue** and waits for the next turn to complete the execution.



Problem Statement :-

" Dr. Shashank Sir 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. "

Solution :-

The given problem is scheduling problem. The problem can be solved by Round Robin algorithm.

Program execution sequence:

1. Taking inputs of queries from user
2. Sorting all queries according to Arrival-Time
3. Merging all queries (initial priority to Faculty's query)
4. Applying Round-Robin algorithm on merged queries
5. Print the result

Steps to follow to execute the program:-

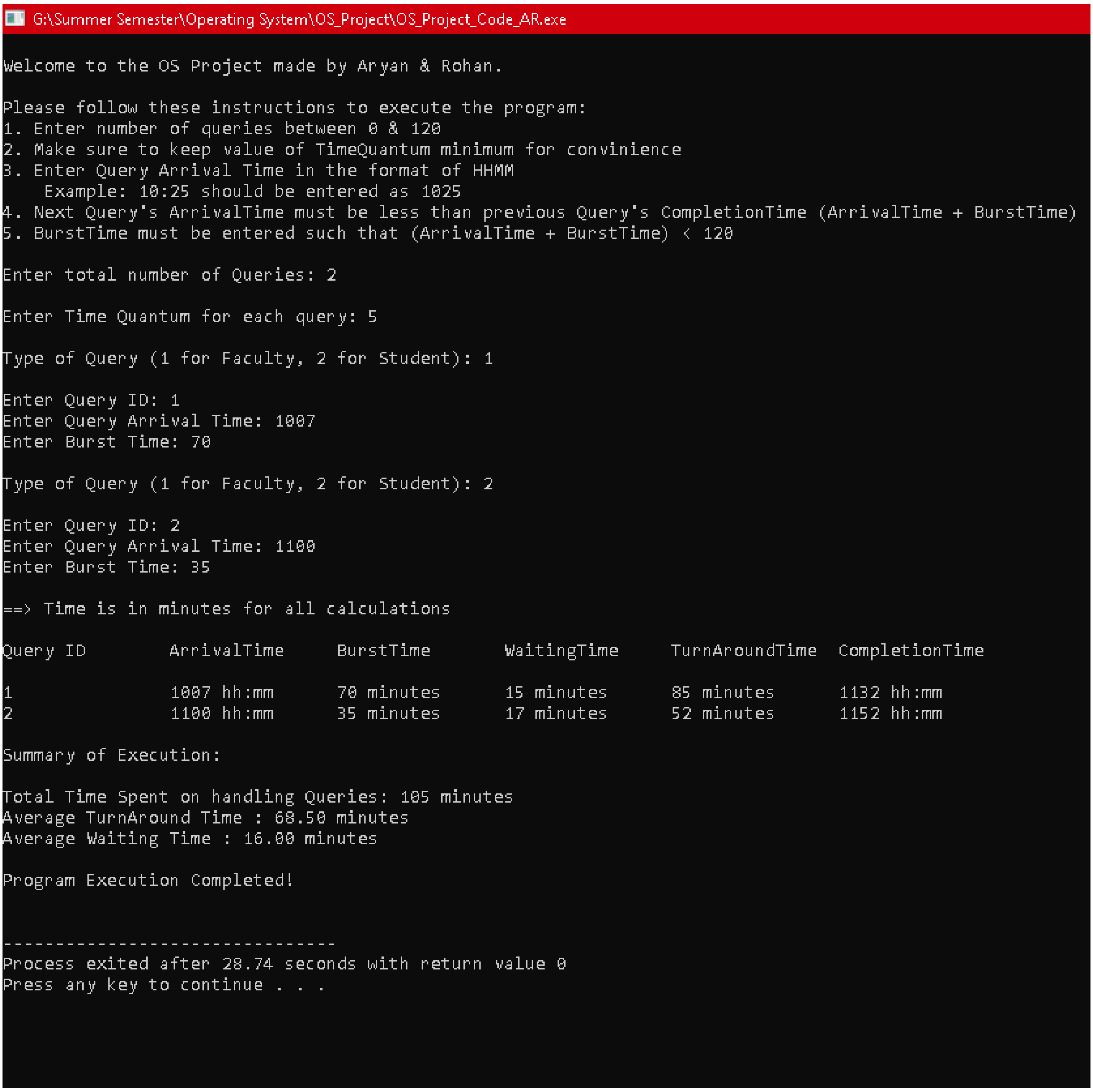
1. Enter number of queries between 0 & 120
2. Make sure to keep value of Time-Quantum minimum
3. Enter Query Arrival Time in the format of HHMM

   Example: 10:25 should be entered as 1025
4. Next Query's Arrival-Time must be less than previous Query's Completion-Time

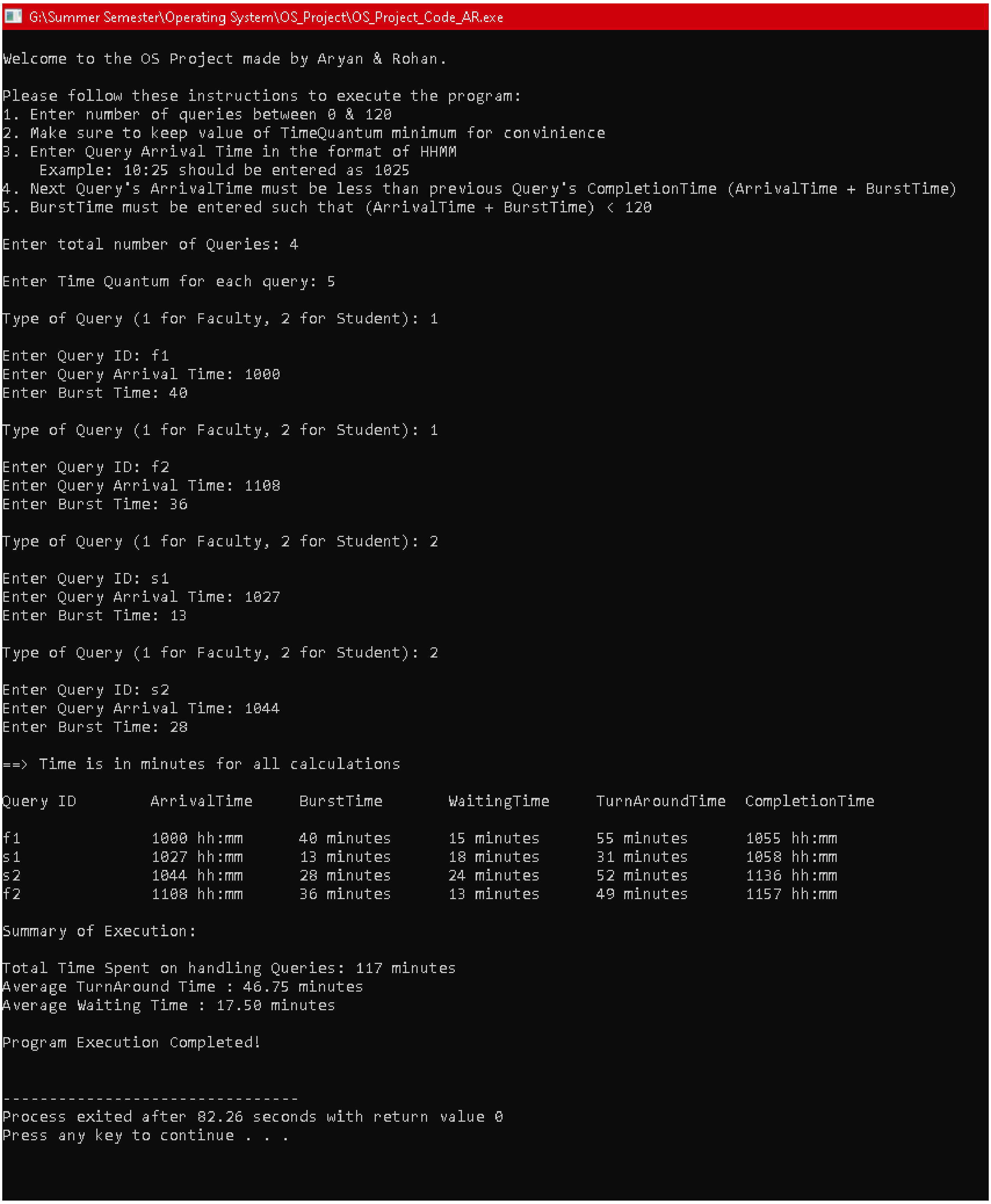
(Arrival-Time + Burst-Time)

5. Burst-Time must be entered such that (Arrival-Time + Burst-Time) < 120

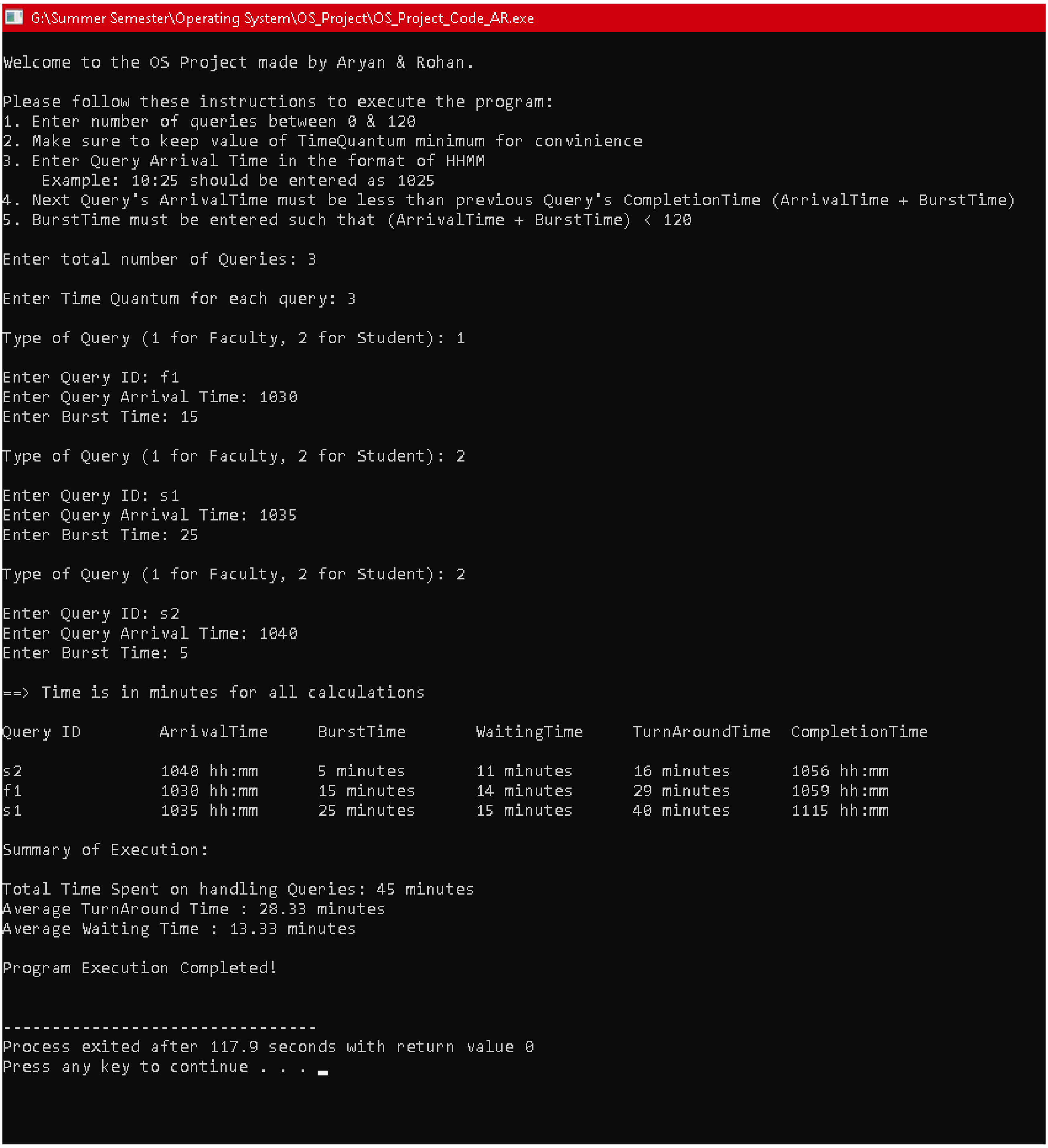
Snap-Shots Of Output :-


G:\Summer Semester\Operating System\OS_Project\OS_Project_Code_AR.exe

```
Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:
1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM
    Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 2

Enter Time Quantum for each query: 5

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: 1
Enter Query Arrival Time: 1007
Enter Burst Time: 70

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: 2
Enter Query Arrival Time: 1100
Enter Burst Time: 35

==> Time is in minutes for all calculations

Query ID        ArrivalTime     BurstTime       WaitingTime     TurnAroundTime  CompletionTime

1               1007 hh:mm      70 minutes      15 minutes      85 minutes      1132 hh:mm
2               1100 hh:mm      35 minutes      17 minutes      52 minutes      1152 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 105 minutes
Average TurnAround Time : 68.50 minutes
Average Waiting Time : 16.00 minutes

Program Execution Completed!


--------------------------------
Process exited after 28.74 seconds with return value 0
Press any key to continue . . .
```

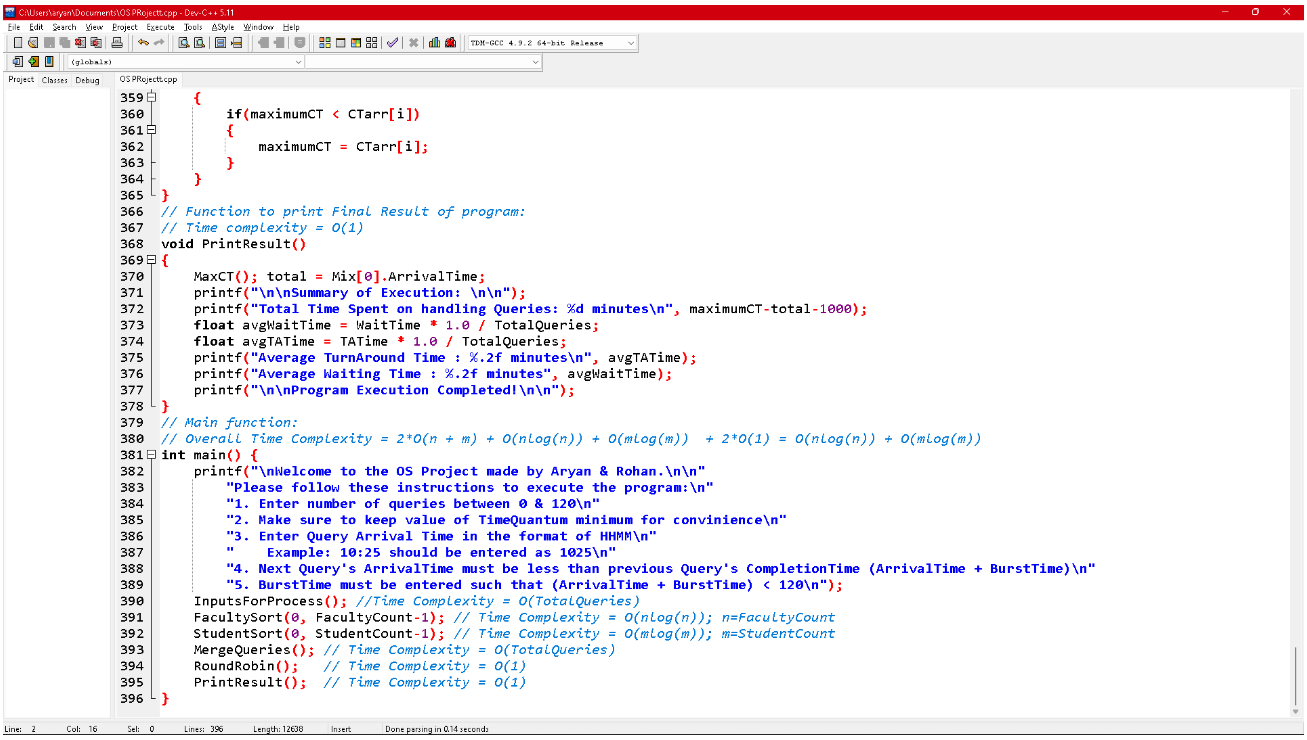

G:\Summer Semester\Operating System\OS_Project\OS_Project_Code_AR.exe

```
Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:
1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM
    Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 4

Enter Time Quantum for each query: 5

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f1
Enter Query Arrival Time: 1000
Enter Burst Time: 40

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f2
Enter Query Arrival Time: 1108
Enter Burst Time: 36

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s1
Enter Query Arrival Time: 1027
Enter Burst Time: 13

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s2
Enter Query Arrival Time: 1044
Enter Burst Time: 28

==> Time is in minutes for all calculations

Query ID        ArrivalTime     BurstTime       WaitingTime     TurnAroundTime  CompletionTime

f1              1000 hh:mm      40 minutes      15 minutes      55 minutes      1055 hh:mm
s1              1027 hh:mm      13 minutes      18 minutes      31 minutes      1058 hh:mm
s2              1044 hh:mm      28 minutes      24 minutes      52 minutes      1136 hh:mm
f2              1108 hh:mm      36 minutes      13 minutes      49 minutes      1157 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 117 minutes
Average TurnAround Time : 46.75 minutes
Average Waiting Time : 17.50 minutes

Program Execution Completed!


--------------------------------
Process exited after 82.26 seconds with return value 0
Press any key to continue . . .
```

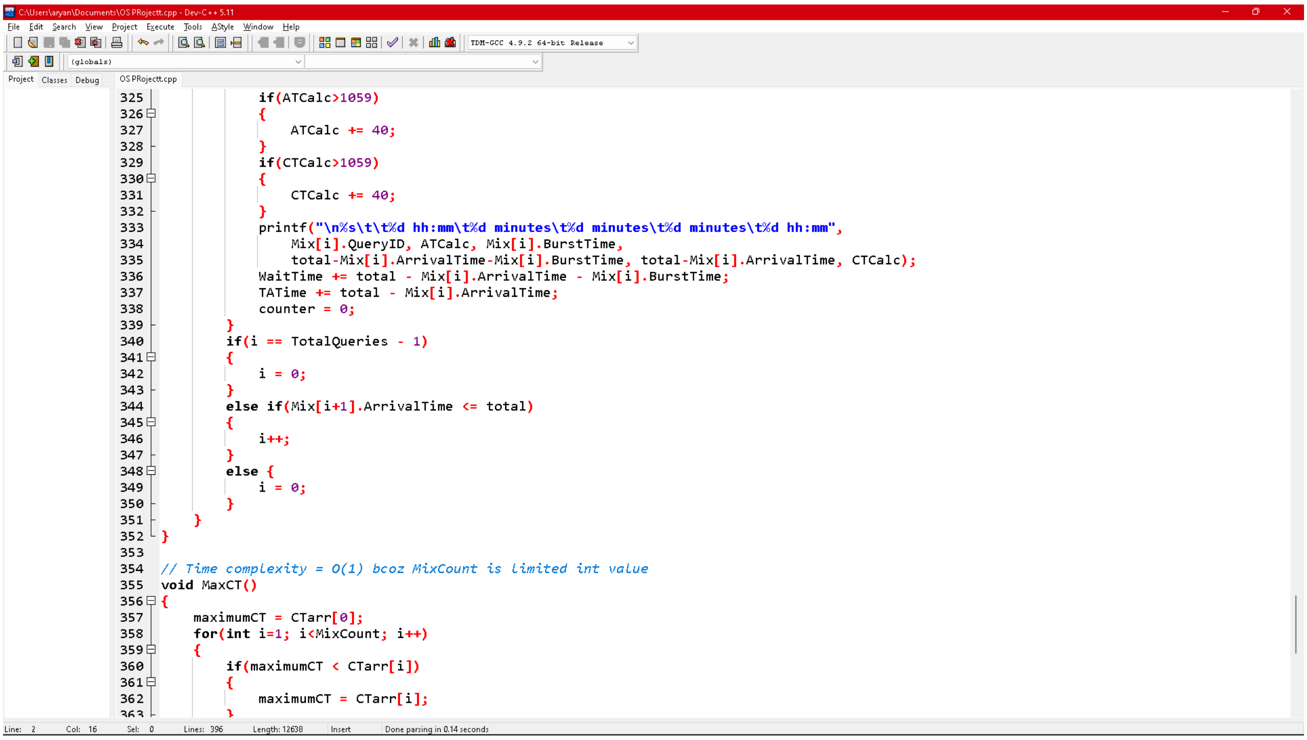

G:\Summer Semester\Operating System\OS_Project\OS_Project_Code_AR.exe

```
Welcome to the OS Project made by Aryan & Rohan.

Please follow these instructions to execute the program:
1. Enter number of queries between 0 & 120
2. Make sure to keep value of TimeQuantum minimum for convinience
3. Enter Query Arrival Time in the format of HHMM
    Example: 10:25 should be entered as 1025
4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)
5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120

Enter total number of Queries: 3

Enter Time Quantum for each query: 3

Type of Query (1 for Faculty, 2 for Student): 1

Enter Query ID: f1
Enter Query Arrival Time: 1030
Enter Burst Time: 15

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s1
Enter Query Arrival Time: 1035
Enter Burst Time: 25

Type of Query (1 for Faculty, 2 for Student): 2

Enter Query ID: s2
Enter Query Arrival Time: 1040
Enter Burst Time: 5

==> Time is in minutes for all calculations

Query ID        ArrivalTime     BurstTime       WaitingTime     TurnAroundTime  CompletionTime

s2              1040 hh:mm      5 minutes       11 minutes      16 minutes      1056 hh:mm
f1              1030 hh:mm      15 minutes      14 minutes      29 minutes      1059 hh:mm
s1              1035 hh:mm      25 minutes      15 minutes      40 minutes      1115 hh:mm

Summary of Execution:

Total Time Spent on handling Queries: 45 minutes
Average TurnAround Time : 28.33 minutes
Average Waiting Time : 13.33 minutes

Program Execution Completed!


--------------------------------
Process exited after 117.9 seconds with return value 0
Press any key to continue . . .
```

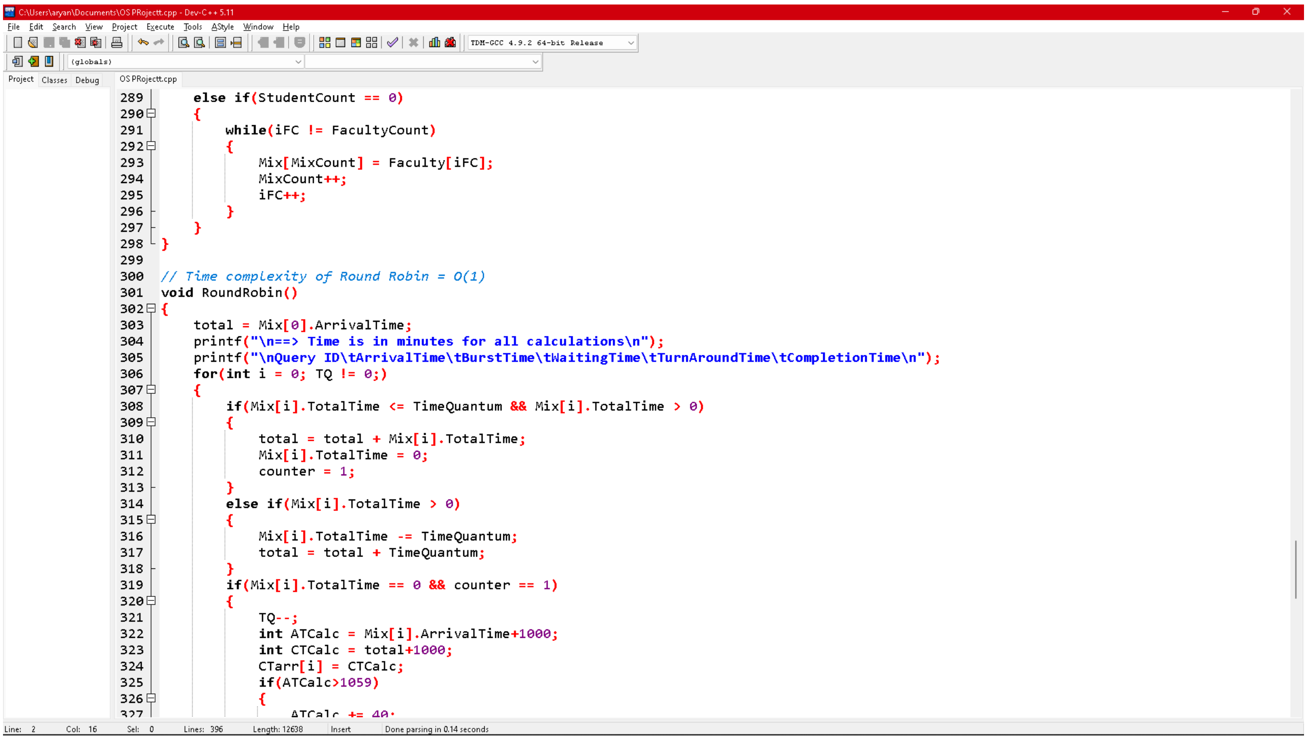
Snap-Shots Of Code:-


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

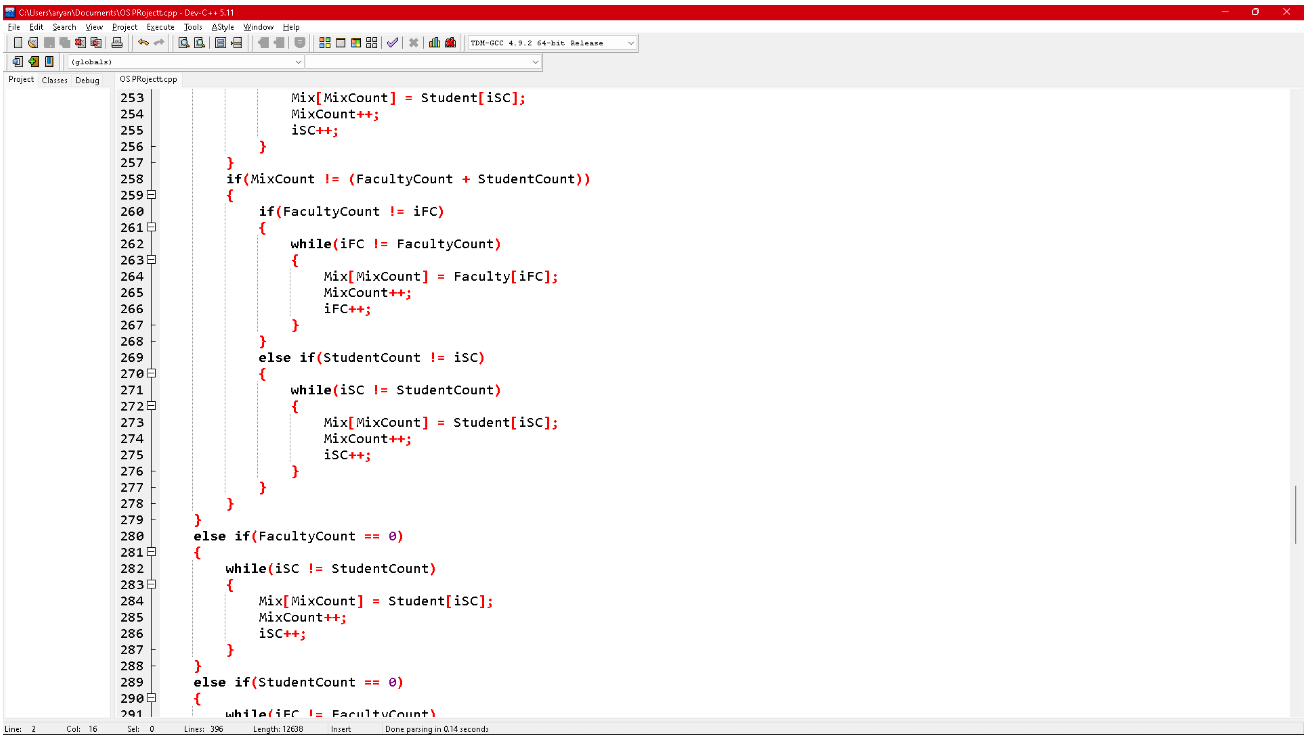
```c
    {
        if(maximumCT < CTarr[i])
        {
            maximumCT = CTarr[i];
        }
    }
}
// Function to print Final Result of program:
// Time complexity = O(1)
void PrintResult()
{
    MaxCT(); total = Mix[0].ArrivalTime;
    printf("\n\nSummary of Execution: \n\n");
    printf("Total Time Spent on handling Queries: %d minutes\n", maximumCT-total-1000);
    float avgWaitTime = WaitTime * 1.0 / TotalQueries;
    float avgTATime = TATime * 1.0 / TotalQueries;
    printf("Average TurnAround Time : %.2f minutes\n", avgTATime);
    printf("Average Waiting Time : %.2f minutes", avgWaitTime);
    printf("\n\nProgram Execution Completed!\n\n");
}
// Main function:
// Overall Time Complexity = 2*O(n + m) + O(nlog(n)) + O(mlog(m))  + 2*O(1) = O(nlog(n)) + O(mlog(m))
int main() {
    printf("\nWelcome to the OS Project made by Aryan & Rohan.\n\n"
        "Please follow these instructions to execute the program:\n"
        "1. Enter number of queries between 0 & 120\n"
        "2. Make sure to keep value of TimeQuantum minimum for convinience\n"
        "3. Enter Query Arrival Time in the format of HHMM\n"
        "    Example: 10:25 should be entered as 1025\n"
        "4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)\n"
        "5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120\n");
    InputsForProcess(); //Time Complexity = O(TotalQueries)
    FacultySort(0, FacultyCount-1); // Time Complexity = O(nlog(n)); n=FacultyCount
    StudentSort(0, StudentCount-1); // Time Complexity = O(mlog(m)); m=StudentCount
    MergeQueries(); // Time Complexity = O(TotalQueries)
    RoundRobin();   // Time Complexity = O(1)
    PrintResult();  // Time Complexity = O(1)
}
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

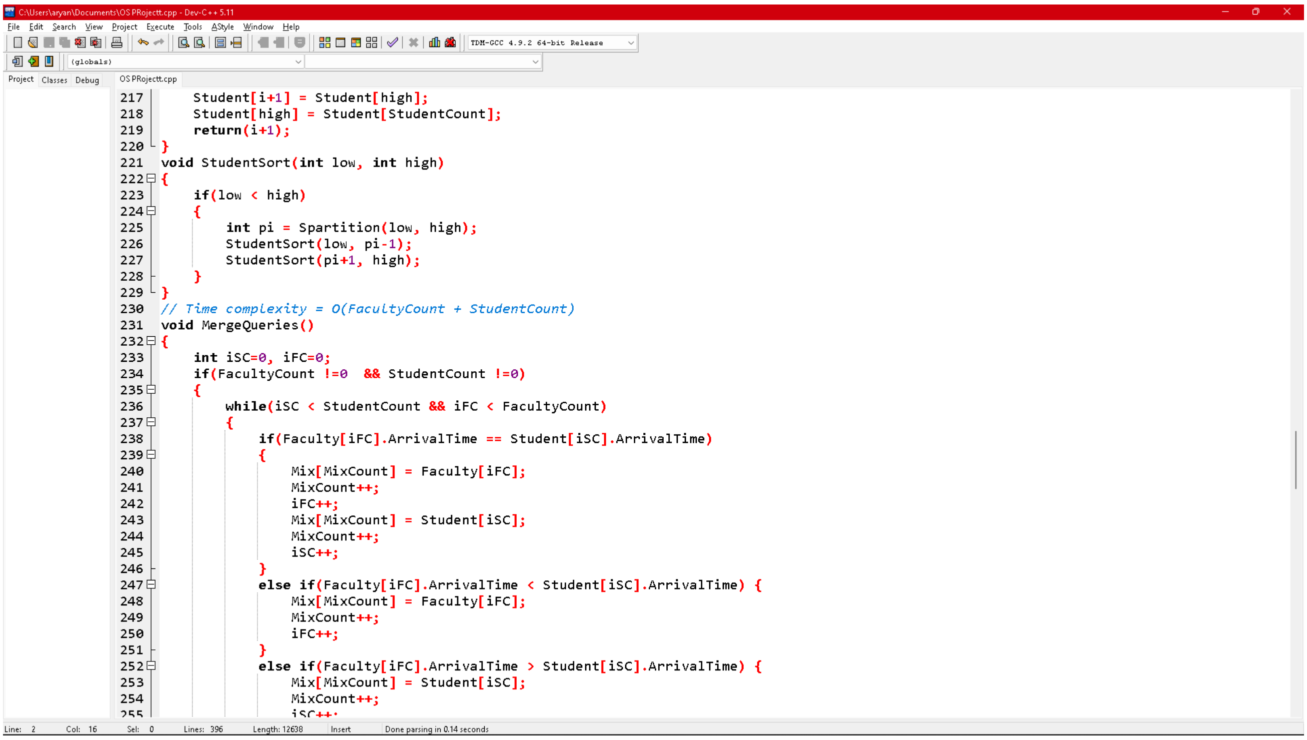
```c
            if(ATCalc>1059)
            {
                ATCalc += 40;
            }
            if(CTCalc>1059)
            {
                CTCalc += 40;
            }
            printf("\n%s\t\t%d hh:mm\t%d minutes\t%d minutes\t%d hh:mm",
                Mix[i].QueryID, ATCalc, Mix[i].BurstTime,
                total-Mix[i].ArrivalTime-Mix[i].BurstTime, total-Mix[i].ArrivalTime, CTCalc);
            WaitTime += total - Mix[i].ArrivalTime - Mix[i].BurstTime;
            TATime += total - Mix[i].ArrivalTime;
            counter = 0;
        }
        if(i == TotalQueries - 1)
        {
            i = 0;
        }
        else if(Mix[i+1].ArrivalTime <= total)
        {
            i++;
        }
        else {
            i = 0;
        }
    }
}

// Time complexity = O(1) bcoz MixCount is limited int value
void MaxCT()
{
    maximumCT = CTarr[0];
    for(int i=1; i<MixCount; i++)
    {
        if(maximumCT < CTarr[i])
        {
            maximumCT = CTarr[i];
        }
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

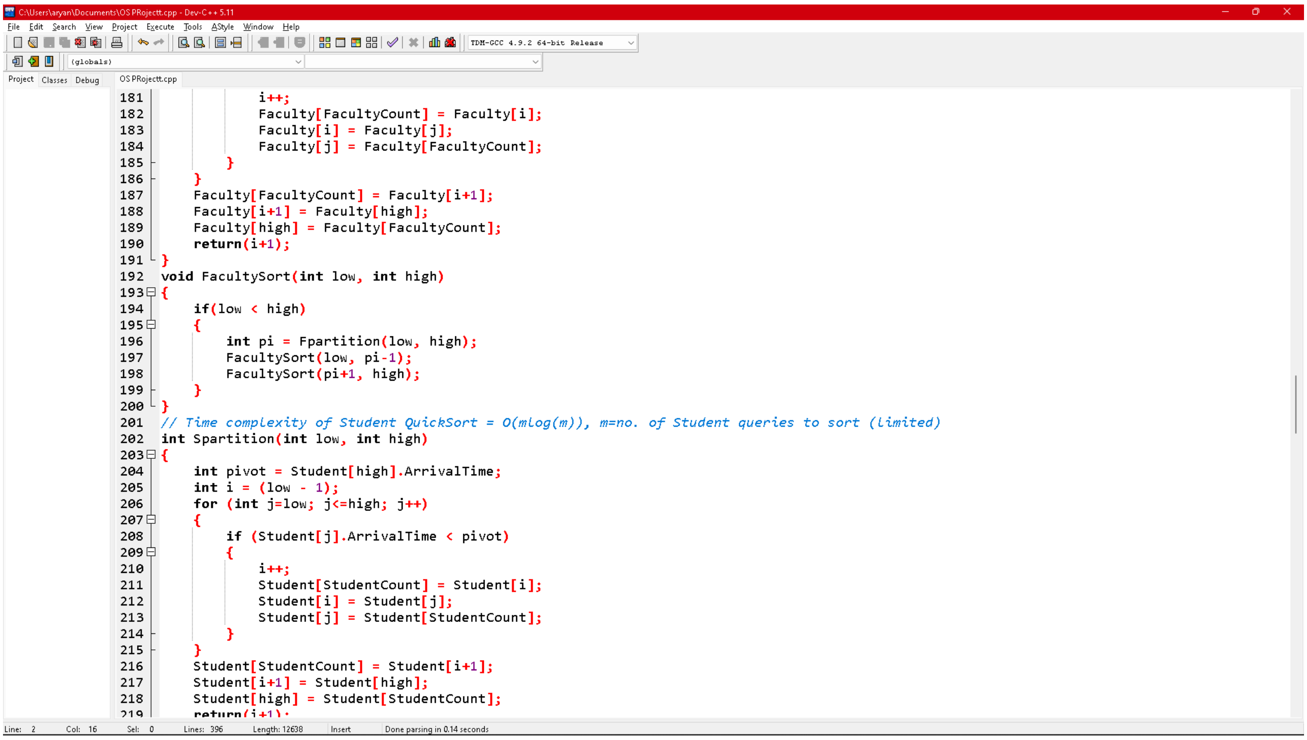
```
    else if(StudentCount == 0)
    {
        while(iFC != FacultyCount)
        {
            Mix[MixCount] = Faculty[iFC];
            MixCount++;
            iFC++;
        }
    }
}

// Time complexity of Round Robin = O(1)
void RoundRobin()
{
    total = Mix[0].ArrivalTime;
    printf("\n==> Time is in minutes for all calculations\n");
    printf("\nQuery ID\tArrivalTime\tBurstTime\tWaitingTime\tTurnAroundTime\tCompletionTime\n");
    for(int i = 0; TQ != 0;)
    {
        if(Mix[i].TotalTime <= TimeQuantum && Mix[i].TotalTime > 0)
        {
            total = total + Mix[i].TotalTime;
            Mix[i].TotalTime = 0;
            counter = 1;
        }
        else if(Mix[i].TotalTime > 0)
        {
            Mix[i].TotalTime -= TimeQuantum;
            total = total + TimeQuantum;
        }
        if(Mix[i].TotalTime == 0 && counter == 1)
        {
            TQ--;
            int ATCalc = Mix[i].ArrivalTime+1000;
            int CTCalc = total+1000;
            CTarr[i] = CTCalc;
            if(ATCalc>1059)
            {
                ATCalc += 40;
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

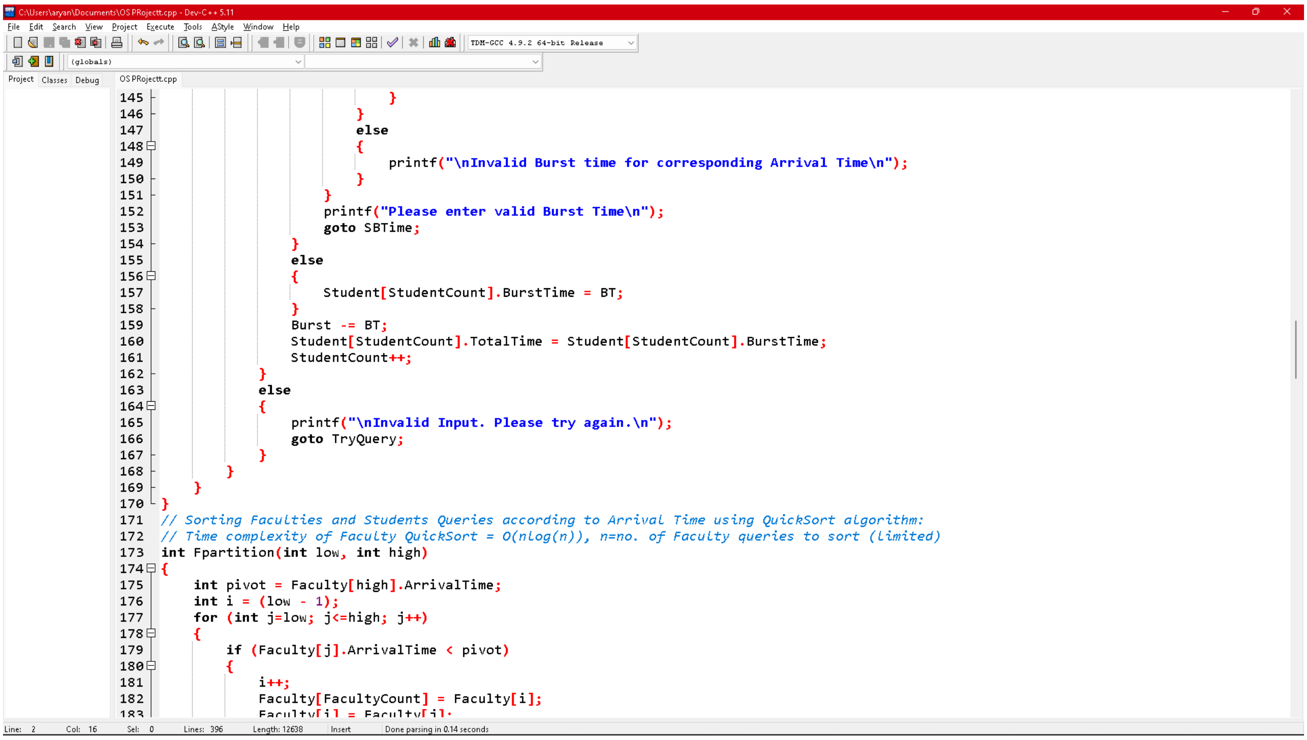
```
                Mix[MixCount] = Student[iSC];
                MixCount++;
                iSC++;
            }
        }
        if(MixCount != (FacultyCount + StudentCount))
        {
            if(FacultyCount != iFC)
            {
                while(iFC != FacultyCount)
                {
                    Mix[MixCount] = Faculty[iFC];
                    MixCount++;
                    iFC++;
                }
            }
            else if(StudentCount != iSC)
            {
                while(iSC != StudentCount)
                {
                    Mix[MixCount] = Student[iSC];
                    MixCount++;
                    iSC++;
                }
            }
        }
    }
    else if(FacultyCount == 0)
    {
        while(iSC != StudentCount)
        {
            Mix[MixCount] = Student[iSC];
            MixCount++;
            iSC++;
        }
    }
    else if(StudentCount == 0)
    {
        while(iFC != FacultyCount)
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

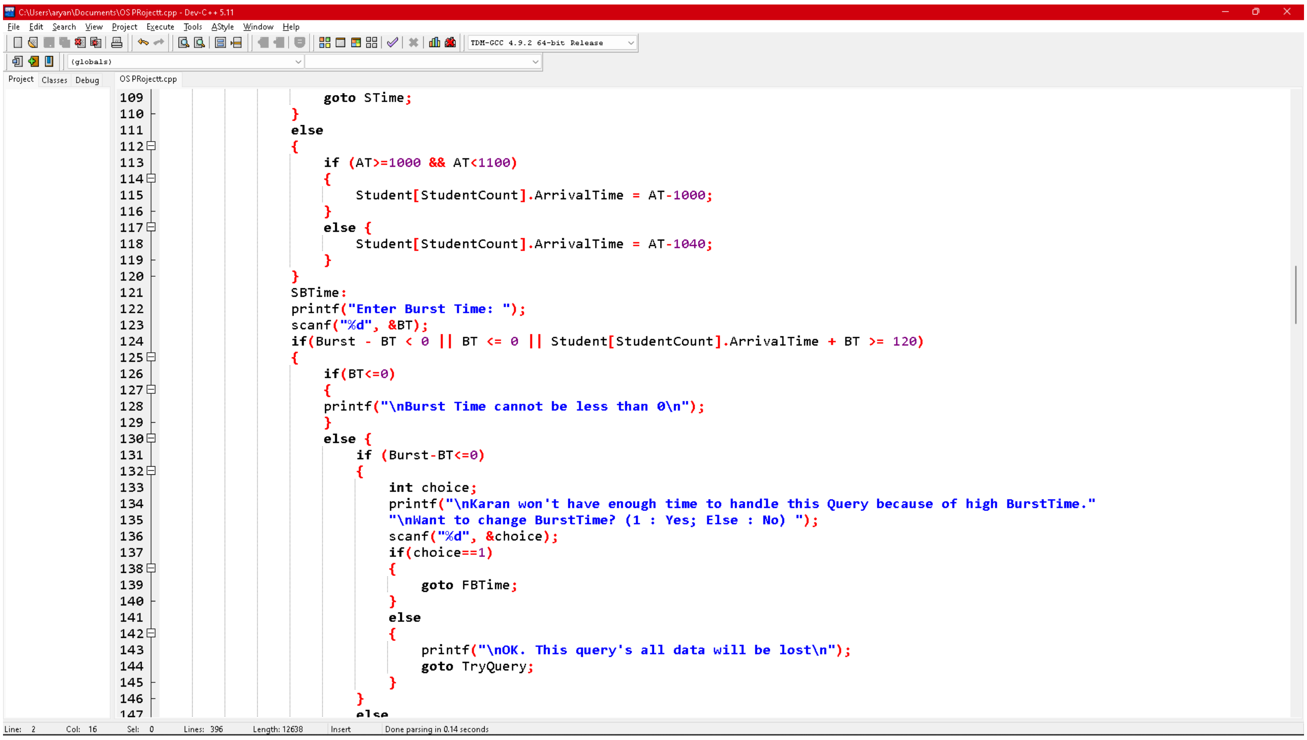
```cpp
    Student[i+1] = Student[high];
    Student[high] = Student[StudentCount];
    return(i+1);
}
void StudentSort(int low, int high)
{
    if(low < high)
    {
        int pi = Spartition(low, high);
        StudentSort(low, pi-1);
        StudentSort(pi+1, high);
    }
}
// Time complexity = O(FacultyCount + StudentCount)
void MergeQueries()
{
    int iSC=0, iFC=0;
    if(FacultyCount !=0  && StudentCount !=0)
    {
        while(iSC < StudentCount && iFC < FacultyCount)
        {
            if(Faculty[iFC].ArrivalTime == Student[iSC].ArrivalTime)
            {
                Mix[MixCount] = Faculty[iFC];
                MixCount++;
                iFC++;
                Mix[MixCount] = Student[iSC];
                MixCount++;
                iSC++;
            }
            else if(Faculty[iFC].ArrivalTime < Student[iSC].ArrivalTime) {
                Mix[MixCount] = Faculty[iFC];
                MixCount++;
                iFC++;
            }
            else if(Faculty[iFC].ArrivalTime > Student[iSC].ArrivalTime) {
                Mix[MixCount] = Student[iSC];
                MixCount++;
                iSC++;
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

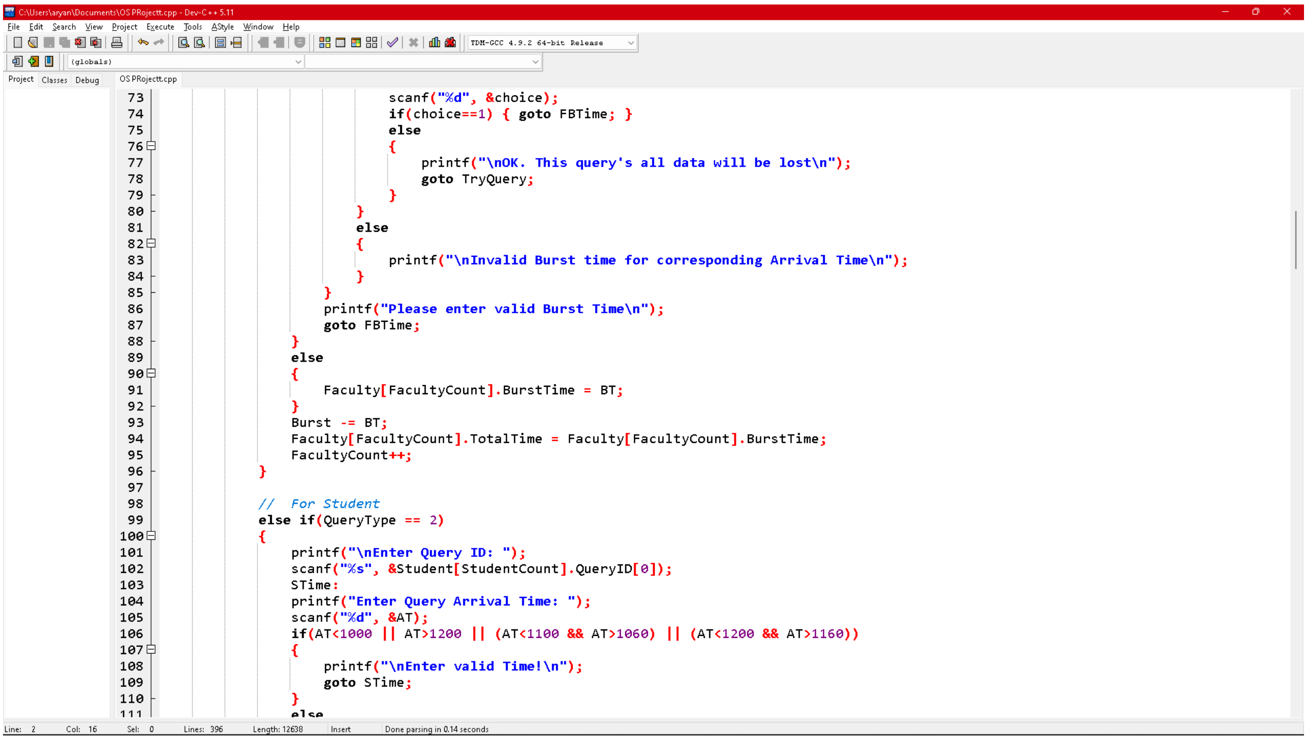
```cpp
            i++;
            Faculty[FacultyCount] = Faculty[i];
            Faculty[i] = Faculty[j];
            Faculty[j] = Faculty[FacultyCount];
        }
    }
    Faculty[FacultyCount] = Faculty[i+1];
    Faculty[i+1] = Faculty[high];
    Faculty[high] = Faculty[FacultyCount];
    return(i+1);
}
void FacultySort(int low, int high)
{
    if(low < high)
    {
        int pi = Fpartition(low, high);
        FacultySort(low, pi-1);
        FacultySort(pi+1, high);
    }
}
// Time complexity of Student QuickSort = O(mlog(m)), m=no. of Student queries to sort (limited)
int Spartition(int low, int high)
{
    int pivot = Student[high].ArrivalTime;
    int i = (low - 1);
    for (int j=low; j<=high; j++)
    {
        if (Student[j].ArrivalTime < pivot)
        {
            i++;
            Student[StudentCount] = Student[i];
            Student[i] = Student[j];
            Student[j] = Student[StudentCount];
        }
    }
    Student[StudentCount] = Student[i+1];
    Student[i+1] = Student[high];
    Student[high] = Student[StudentCount];
    return(i+1);
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

```
                            }
                        }
                        else
                        {
                            printf("\nInvalid Burst time for corresponding Arrival Time\n");
                        }
                    }
                    printf("Please enter valid Burst Time\n");
                    goto SBTime;
                }
                else
                {
                    Student[StudentCount].BurstTime = BT;
                }
                Burst -= BT;
                Student[StudentCount].TotalTime = Student[StudentCount].BurstTime;
                StudentCount++;
            }
            else
            {
                printf("\nInvalid Input. Please try again.\n");
                goto TryQuery;
            }
        }
    }
}
// Sorting Faculties and Students Queries according to Arrival Time using QuickSort algorithm:
// Time complexity of Faculty QuickSort = O(nlog(n)), n=no. of Faculty queries to sort (Limited)
int Fpartition(int low, int high)
{
    int pivot = Faculty[high].ArrivalTime;
    int i = (low - 1);
    for (int j=low; j<=high; j++)
    {
        if (Faculty[j].ArrivalTime < pivot)
        {
            i++;
            Faculty[FacultyCount] = Faculty[i];
            Faculty[i] = Faculty[j];
```

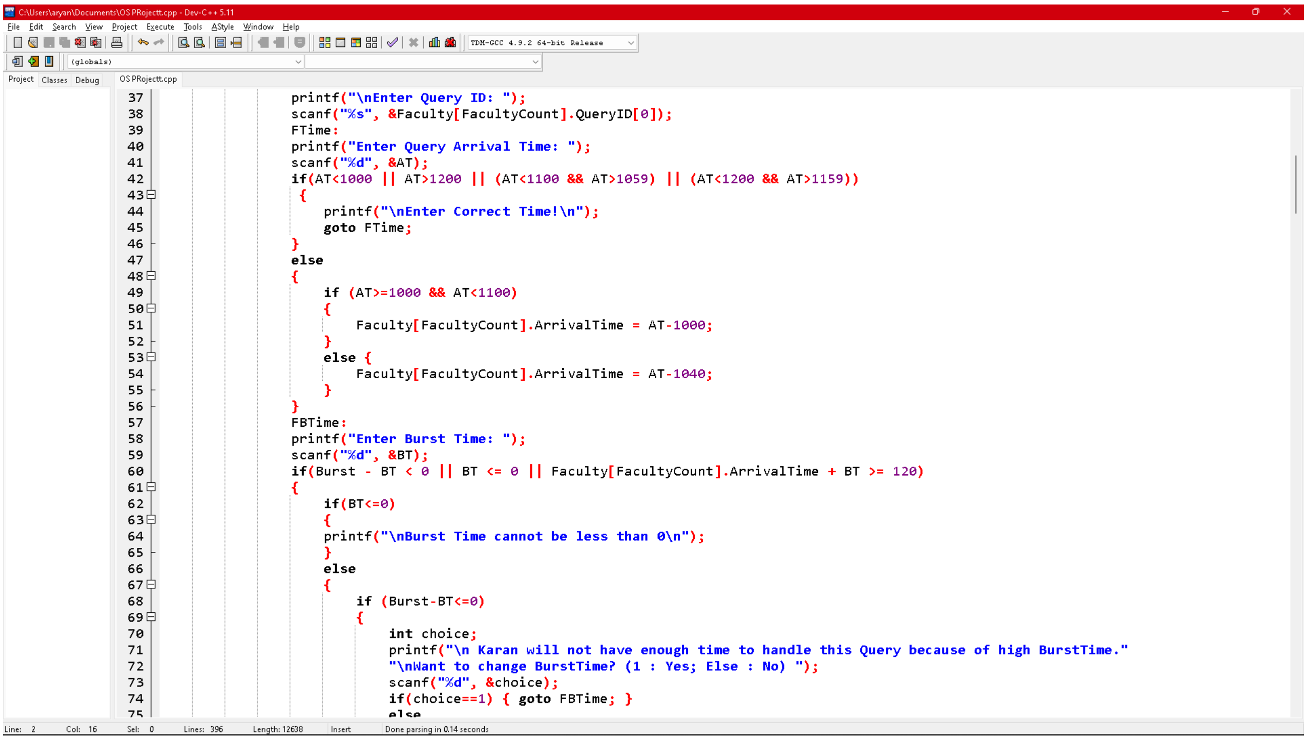

C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

```
                    goto STime;
                }
                else
                {
                    if (AT>=1000 && AT<1100)
                    {
                        Student[StudentCount].ArrivalTime = AT-1000;
                    }
                    else {
                        Student[StudentCount].ArrivalTime = AT-1040;
                    }
                }
                SBTime:
                printf("Enter Burst Time: ");
                scanf("%d", &BT);
                if(Burst - BT < 0 || BT <= 0 || Student[StudentCount].ArrivalTime + BT >= 120)
                {
                    if(BT<=0)
                    {
                    printf("\nBurst Time cannot be less than 0\n");
                    }
                    else {
                        if (Burst-BT<=0)
                        {
                            int choice;
                            printf("\nKaran won't have enough time to handle this Query because of high BurstTime."
                            "\nWant to change BurstTime? (1 : Yes; Else : No) ");
                            scanf("%d", &choice);
                            if(choice==1)
                            {
                                goto FBTime;
                            }
                            else
                            {
                                printf("\nOK. This query's all data will be lost\n");
                                goto TryQuery;
                            }
                        }
                        else
```


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp

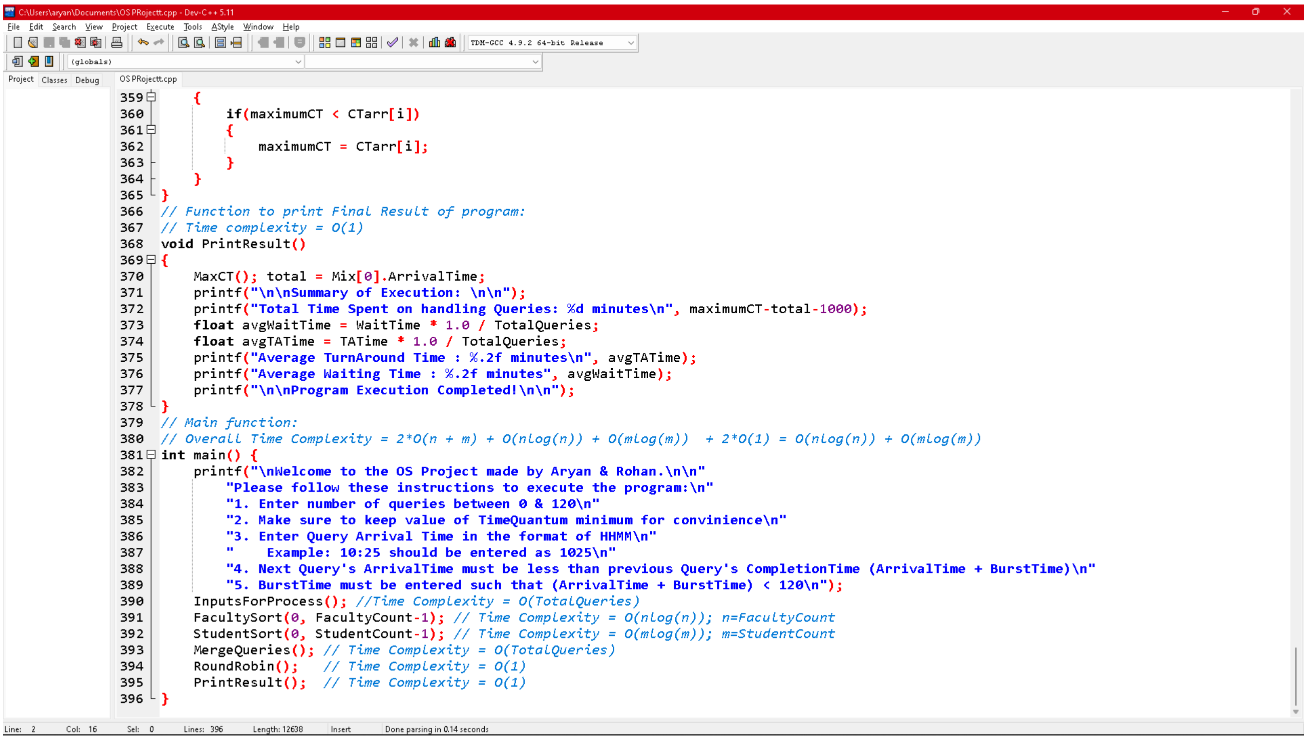
```
                            scanf("%d", &choice);
                            if(choice==1) { goto FBTime; }
                            else
                            {
                                printf("\nOK. This query's all data will be lost\n");
                                goto TryQuery;
                            }
                        }
                        else
                        {
                            printf("\nInvalid Burst time for corresponding Arrival Time\n");
                        }
                    }
                    printf("Please enter valid Burst Time\n");
                    goto FBTime;
                }
                else
                {
                    Faculty[FacultyCount].BurstTime = BT;
                }
                Burst -= BT;
                Faculty[FacultyCount].TotalTime = Faculty[FacultyCount].BurstTime;
                FacultyCount++;
            }

            //  For Student
            else if(QueryType == 2)
            {
                printf("\nEnter Query ID: ");
                scanf("%s", &Student[StudentCount].QueryID[0]);
                STime:
                printf("Enter Query Arrival Time: ");
                scanf("%d", &AT);
                if(AT<1000 || AT>1200 || (AT<1100 && AT>1060) || (AT<1200 && AT>1160))
                {
                    printf("\nEnter valid Time!\n");
                    goto STime;
                }
                else
```

Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp

```
                printf("\nEnter Query ID: ");
                scanf("%s", &Faculty[FacultyCount].QueryID[0]);
                FTime:
                printf("Enter Query Arrival Time: ");
                scanf("%d", &AT);
                if(AT<1000 || AT>1200 || (AT<1100 && AT>1059) || (AT<1200 && AT>1159))
                 {
                    printf("\nEnter Correct Time!\n");
                    goto FTime;
                }
                else
                {
                    if (AT>=1000 && AT<1100)
                    {
                        Faculty[FacultyCount].ArrivalTime = AT-1000;
                    }
                    else {
                        Faculty[FacultyCount].ArrivalTime = AT-1040;
                    }
                }
                FBTime:
                printf("Enter Burst Time: ");
                scanf("%d", &BT);
                if(Burst - BT < 0 || BT <= 0 || Faculty[FacultyCount].ArrivalTime + BT >= 120)
                {
                    if(BT<=0)
                    {
                    printf("\nBurst Time cannot be less than 0\n");
                    }
                    else
                    {
                        if (Burst-BT<=0)
                        {
                            int choice;
                            printf("\n Karan will not have enough time to handle this Query because of high BurstTime."
                            "\nWant to change BurstTime? (1 : Yes; Else : No) ");
                            scanf("%d", &choice);
                            if(choice==1) { goto FBTime; }
                            else
```

Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds


C:\Users\aryan\Documents\OS PRojectt.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
OS PRojectt.cpp
Line: 2 Col: 16 Sel: 0 Lines: 396 Length: 12638 Insert Done parsing in 0.14 seconds

```
    {
        if(maximumCT < CTarr[i])
        {
            maximumCT = CTarr[i];
        }
    }
}
// Function to print Final Result of program:
// Time complexity = O(1)
void PrintResult()
{
    MaxCT(); total = Mix[0].ArrivalTime;
    printf("\n\nSummary of Execution: \n\n");
    printf("Total Time Spent on handling Queries: %d minutes\n", maximumCT-total-1000);
    float avgWaitTime = WaitTime * 1.0 / TotalQueries;
    float avgTATime = TATime * 1.0 / TotalQueries;
    printf("Average TurnAround Time : %.2f minutes\n", avgTATime);
    printf("Average Waiting Time : %.2f minutes", avgWaitTime);
    printf("\n\nProgram Execution Completed!\n\n");
}
// Main function:
// Overall Time Complexity = 2*O(n + m) + O(nlog(n)) + O(mlog(m))  + 2*O(1) = O(nlog(n)) + O(mlog(m))
int main() {
    printf("\nWelcome to the OS Project made by Aryan & Rohan.\n\n"
        "Please follow these instructions to execute the program:\n"
        "1. Enter number of queries between 0 & 120\n"
        "2. Make sure to keep value of TimeQuantum minimum for convinience\n"
        "3. Enter Query Arrival Time in the format of HHMM\n"
        "     Example: 10:25 should be entered as 1025\n"
        "4. Next Query's ArrivalTime must be less than previous Query's CompletionTime (ArrivalTime + BurstTime)\n"
        "5. BurstTime must be entered such that (ArrivalTime + BurstTime) < 120\n");
    InputsForProcess(); //Time Complexity = O(TotalQueries)
    FacultySort(0, FacultyCount-1); // Time Complexity = O(nlog(n)); n=FacultyCount
    StudentSort(0, StudentCount-1); // Time Complexity = O(mlog(m)); m=StudentCount
    MergeQueries(); // Time Complexity = O(TotalQueries)
    RoundRobin();   // Time Complexity = O(1)
    PrintResult();  // Time Complexity = O(1)
}
```

Thank You 😊