**Student name:**Karan Bhoker

**Student ID:**11709986

**E-mail** [**address:**](mailto:jyotigarima2000@gmail.com)karanbhoker@gmail.com

**GitHub link:** <https://github.com/Karan0010/OS-final>

**OPERATING SYSTEMS**

**NAME** : Karan Bhoker

**SECTION** : K17UG

**REG ID** : 11709986

**ROLL NO** : A22

**GROUP** : 1

**COURSE CODE** : CSE316

**Student name:**Karan Bhoker

**Student ID:**11709986

**E-mail** [**address:karanbhoker**@gmail.com](mailto:jyotigarima2000@gmail.com)

**GitHub link:** <https://github.com/Karan0010/OS-final>

**Q8.** 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, where faculty queue is given a higher priority. 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.

# ALGORITHMS USED FOR SUPPORTING THE SOLUTION

Round Robin algorithm is used to sort the required querry.

# DESCRIPTION

* Round Robin is a  CPU Scheduling Algorithm where each process is assigned a fixed time slot in a cyclic way. It is simple, easy to implement, and starvation-free as all processes get fair share of CPU. One of the most commonly used technique in CPU scheduling as a core. It is preemptive as processes are assigned CPU only for a fixed slice of time at most. The disadvantage of it is more overhead of context switching.
* **Completion time**: Time at which the process completes its execution
* **Turnaround time**: Time difference between completion time and arrival time
* **Waiting time**: Time difference between turnaround time and burst time
* **Complexity**: (3n)^2

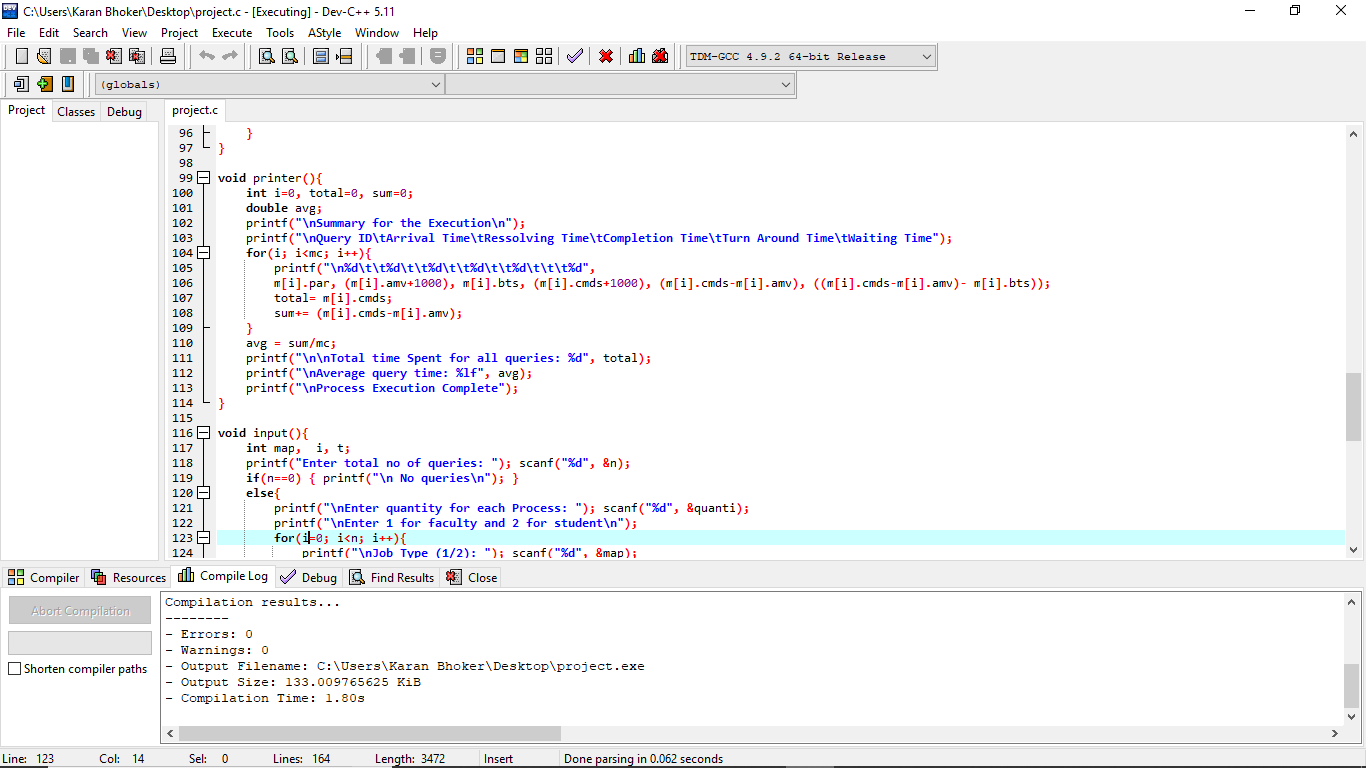
**CONSTRAINTS:**

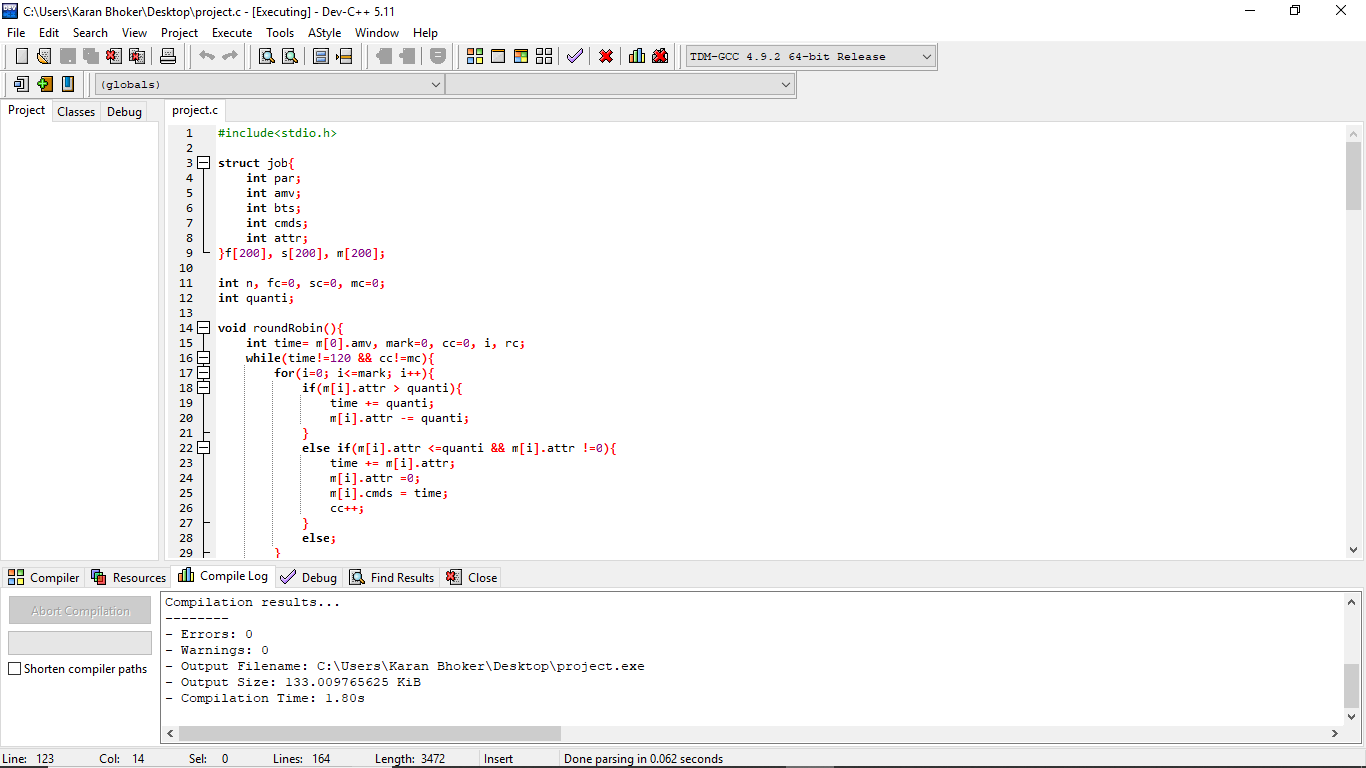
1. Total Waiting Time

2. Average waiting Time

3. Burst Time

4. Turn around Time



****

# CODE:

#include<stdio.h>

struct job{

int par;

int amv;

int bts;

int cmds;

int attr;

}f[200], s[200], m[200];

int n, fc=0, sc=0, mc=0;

int quanti;

void roundRobin(){

int time= m[0].amv, mark=0, cc=0, i, rc;

while(time!=120 && cc!=mc){

for(i=0; i<=mark; i++){

if(m[i].attr > quanti){

time += quanti;

m[i].attr -= quanti;

}

else if(m[i].attr <=quanti && m[i].attr !=0){

time += m[i].attr;

m[i].attr =0;

m[i].cmds = time;

cc++;

}

else;

}

int start = mark+1;

for(rc= start; rc<mc; rc++){

if(m[rc].amv <= time){

mark++;

}

}

}

}

void merger(){

int isc=0, ifc= 0, min, flag;

if( fc!=0 && sc!=0){

while(isc<sc && ifc<fc){

if(f[ifc].amv == s[isc].amv){

m[mc] = f[ifc];

mc++;

ifc++;

m[mc]= s[isc];

mc++;

isc++;

}

else if(f[ifc].amv < s[isc].amv){

m[mc]= f[ifc];

mc++;

ifc++;

}

else if(f[ifc].amv > s[isc].amv){

m[mc]= s[isc];

mc++;

isc++;

}

else;

}

if(mc != (fc+sc)){

if(fc!=ifc){

while(ifc!=fc){

m[mc]= f[ifc];

mc++;

ifc++;

}

}

else if(sc!=isc){

while(isc!=sc){

m[mc]= s[isc];

mc++;

isc++;

}

}

}

}

else if(fc==0){

while(isc!=sc){

m[mc]= s[isc];

mc++;

isc++;

}

}

else if(sc==0){

while(ifc!=fc){

m[mc]= f[ifc];

mc++;

ifc++;

}

}

else {

printf("\n No valid Jobs available\n");

}

}

void printer(){

int i=0, total=0, sum=0;

double avg;

printf("\nSummary for the Execution\n");

printf("\nQuery ID\tArrival Time\tRessolving Time\tCompletion Time\tTurn Around Time\tWaiting Time");

for(i; i<mc; i++){

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d\t\t\t%d",

m[i].par, (m[i].amv+1000), m[i].bts, (m[i].cmds+1000), (m[i].cmds-m[i].amv), ((m[i].cmds-m[i].amv)- m[i].bts));

total= m[i].cmds;

sum+= (m[i].cmds-m[i].amv);

}

avg = sum/mc;

printf("\n\nTotal time Spent for all queries: %d", total);

printf("\nAverage query time: %lf", avg);

printf("\nProcess Execution Complete");

}

void input(){

int map, i, t;

printf("Enter total no of queries: "); scanf("%d", &n);

if(n==0) { printf("\n No queries\n"); }

else{

printf("\nEnter quantity for each Process: "); scanf("%d", &quanti);

printf("\nEnter 1 for faculty and 2 for student\n");

for(i=0; i<n; i++){

printf("\nJob Type (1/2): "); scanf("%d", &map);

if(map==1){

printf("Query Id: "); scanf("%d", &f[fc].par);

printf("Arrival Time: "); scanf("%d", &t);

if(t<1000 || t>1200){

printf("\nEnter Correct time");

input();

}

else{f[fc].amv= t-1000;}

printf("Resolving Time: "); scanf("%d", &f[fc].bts); f[fc].attr= f[fc].bts;

fc++;

} else{

printf("Query Id: "); scanf("%d", &s[sc].par);

printf("Arrival Time: "); scanf("%d", &t);

if(t<1000 || t>1200){

printf("\nEnter Correct time\n");

input();

}

else {s[sc].amv= t-1000; }

printf("Resolving Time: "); scanf("%d", &s[sc].bts); s[sc].attr= s[sc].bts;

sc++;

}

}

}

}

void inst(){

printf("\nWelcome,Please follow these instruction"

"\n\*\*>Enter time in 2400 hours format. example for 11:30 am enter 1130"

"\n\*\*>Enter Query arrival times in ascending order\n"

"\nAll Time units are in minutes. \n\n"

);

}

main(){

inst();

input();

merger();

roundRobin();

printer();

}

**BOUNDARY CONDITIONS:**

1. The Burst time values should be integer.

2. The arrival time should be integer

3. The turn around time should be between 1000 to 1200

4. Burst time should be zero or greater than 0.

5. Job scheduling is only for students and faculty members only.

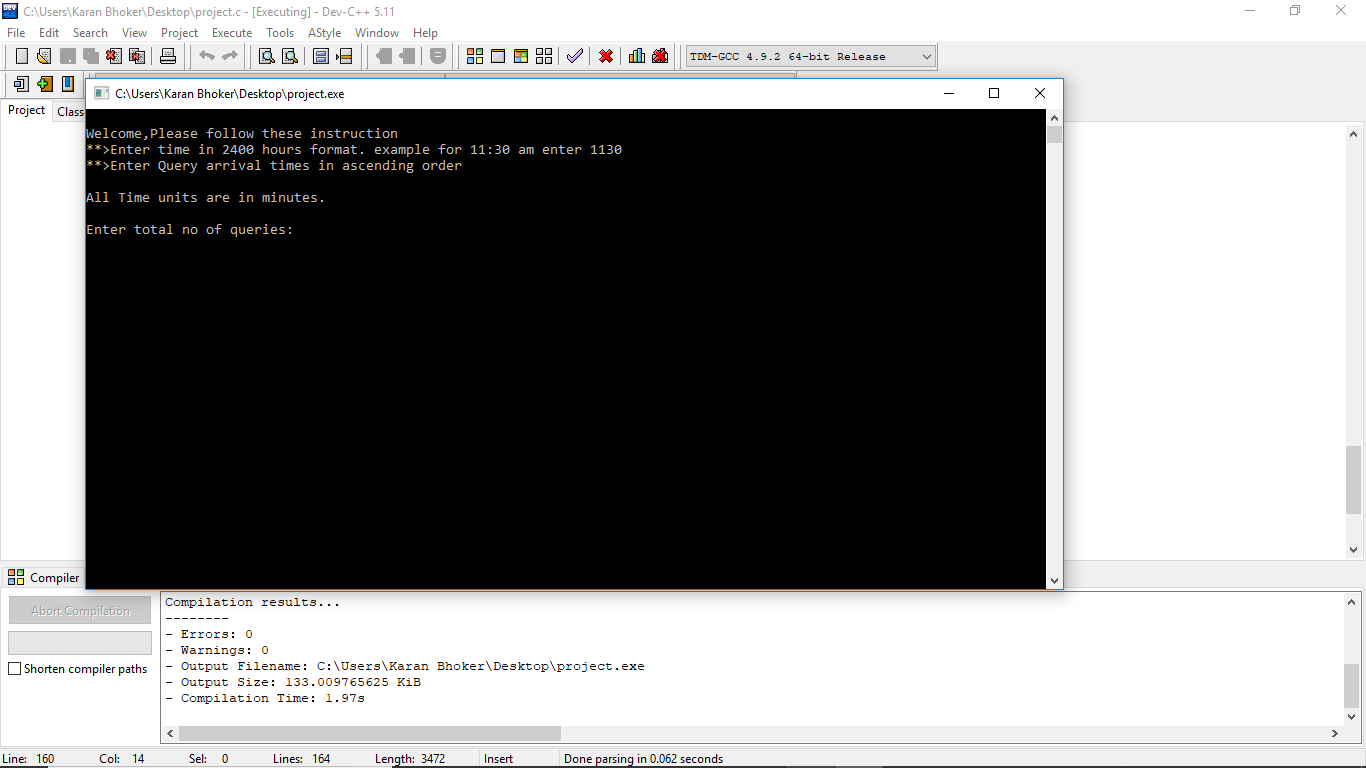
**Student name:**Karan Bhoker

**Student ID:**11709986

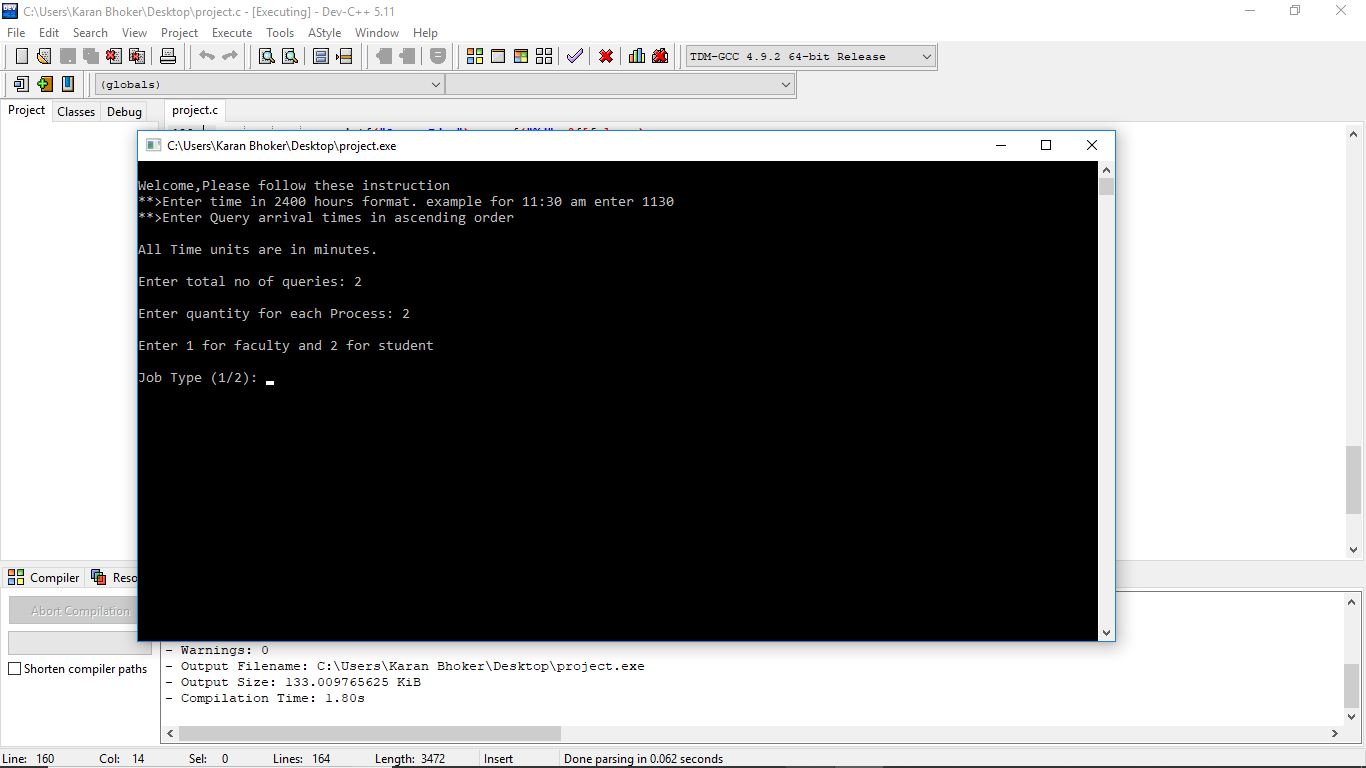
**E-mail** [**address:**karanbhoker@gmail.com](mailto:jyotigarima2000@gmail.com)

**GitHub link:** <https://github.com/Karan0010/OS-final>

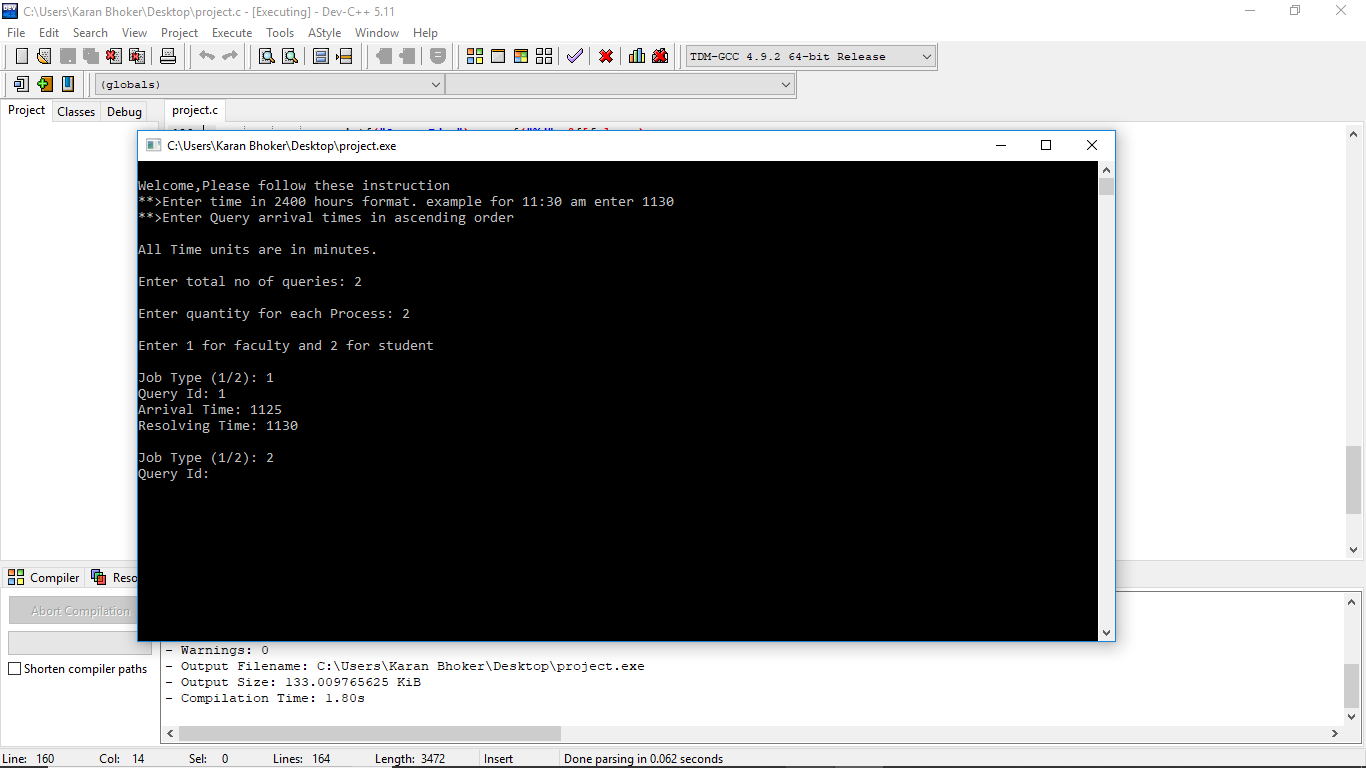
# TEST CASES

Total no of queries

Job Type(Faculty or student)



Querry Id,ArrivalTime,Resolving Time



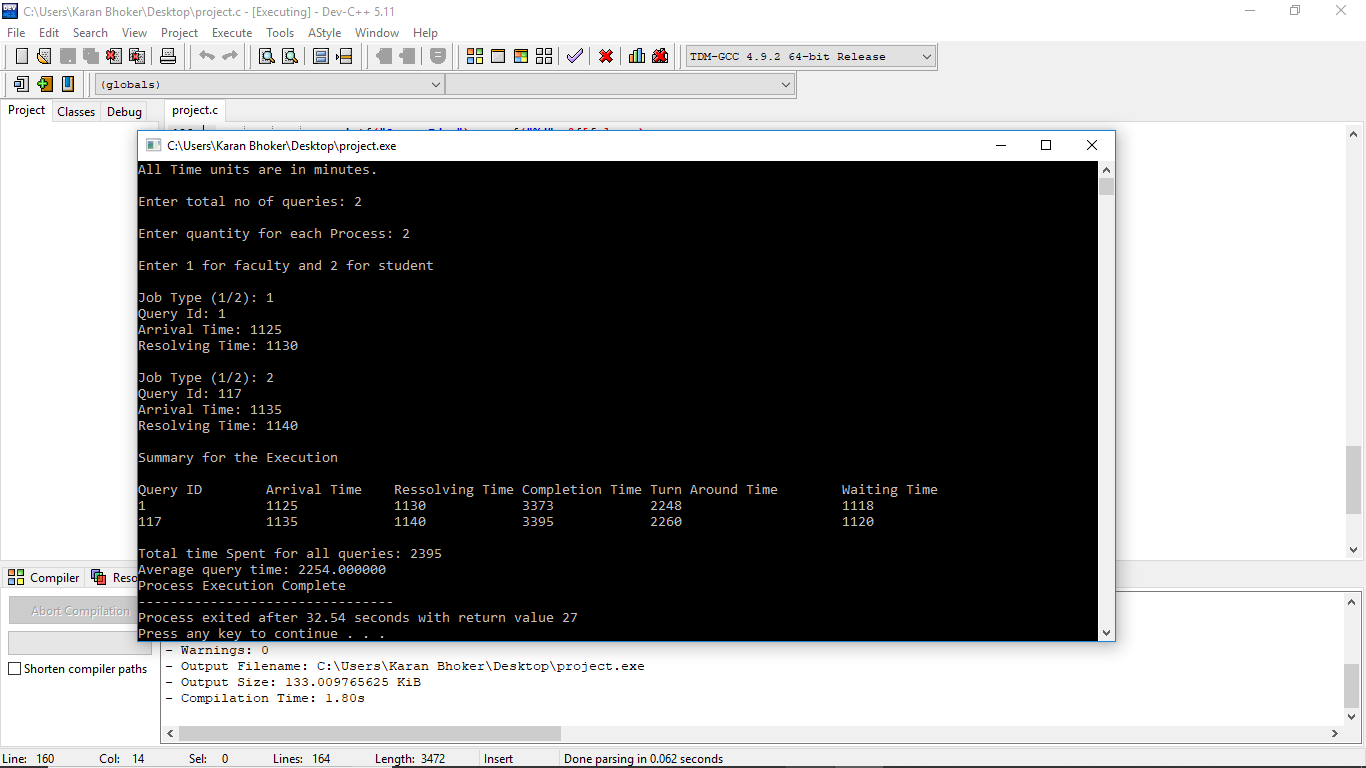
**Student name:**Karan Bhoker

**Student ID:**11709986

**E-mail** [**address:**karanbhoker@gmail.com](mailto:jyotigarima2000@gmail.com)

**GitHub link:** <https://github.com/Karan0010/OS-final>

Summary For the execution



Description:

Yes, I have made minimum of 5 revision but they are a small part of each other itself. First I worked straight on the final project but it was confusing so I divided the work accordingly.

**GitHub Link:** <https://github.com/Karan0010/OS-final>