**Student Name:** Chand Kamal Singh

**Student ID:** 1170426

**Email Address:** ck200599@gmail.com

**GitHub Link:** <https://github.com/CK200/Round_Robin_OS>.git

**Ques. Design a scheduler that can schedule the processes arriving system at periodical intervals. Every process is assigned with a fixed time slice t milliseconds. If it is not able to complete its execution within the assigned time quantum, then automated timer generates an interrupt. The scheduler will select the next process in the queue and dispatcher dispatches the process to processor for execution. Compute the total time for which processes were in the queue waiting for the processor. Take the input for CPU burst, arrival time and time quantum from the user.**

**Required Algorithm:**

1- Create an array mb\_t[] to keep track of remaining

burst time of processes. This array is initially a

copy of bt[] (burst times array)

2- Create another array w\_t[] to store waiting times

of processes. Initialize this array as 0.

3- Initialize time : t = 0

4- Keep traversing the all processes while all processes

are not done. Do following for i'th process if it is

not done yet.

a- If mb\_t[i] > quantum

(i) t = t + quantum

(ii) mb\_t[i] -= quantum;

c- Else // Last cycle for this process

(i) t = t + mb\_t[i];

(ii) w\_t[i] = t - bt[i]

(ii) mb\_t[i] = 0; // This process is over

**Complexity:**

**Complexity=O(n\*n)**

**Boundary condition: -**

1. Number of processes should be positive.
2. Arrival time and Burst time should be greater than 0.

**Solution Code:**

#include<iostream>

#include<stdlib.h>

#include<conio.h>

using namespace std;

struct process //data structure which tracks various variables of process

{

int p\_id;

int a\_t;

int b\_t;

int mb\_t;

int c\_t;

int tat;

int w\_t;

int r\_t;

};

int main()

{

int x,i,t\_q,ct=0,flag=0,min,temp;//variable decleration

cout<<endl<<"--------------------------------------------------ROUND ROBIN SCHEDULER------------------------------------------------"<<endl;

cout<<endl<<"enter the no of processes for scheduling->";

cin>>x; // number of processes

cout<<endl;

process p[x];

for(i=0;i<x;i++) //input from user for process

{

cout<<"enter pid for process->";

cin>>p[i].p\_id;

cout<<"enter arrival time->";

cin>>p[i].a\_t;

if(i==0)

{

min=p[i].a\_t;

}

if(min>p[i].a\_t)

{

min=p[i].a\_t;

}

cout<<"enter burst time->";

cin>>p[i].b\_t;

p[i].mb\_t=p[i].b\_t;

cout<<endl;

}

ct=min;

cout<<"enter the time quantum for scheduler->";

cin>>t\_q; // time quantum for scheduler

while(true) //infinite loop breaks when all process excuted

{

if(flag==x)//breaks loop when all processes are excuted

{

break;

}

flag=0;

for(i=0;i<x;i++)//loop to check all process per time quantum

{

if(p[i].a\_t<=ct && p[i].mb\_t>0)//test condition for scheduling

{

if(p[i].mb\_t==p[i].b\_t)

{

p[i].r\_t=ct;//assigns response time

}

if((p[i].mb\_t-t\_q)>=0)

{

p[i].mb\_t=p[i].mb\_t-t\_q;

ct=ct+t\_q;

}

else

{

ct=ct+p[i].mb\_t;

p[i].mb\_t=0;

}

if(p[i].mb\_t==0)

{

p[i].c\_t=ct;//assigns completion time

}

}

}

for(i=0;i<x;i++)//loop which checks how many processes are excuted

{

if(p[i].mb\_t==0)

{

flag++;

}

}

}

for(i=0;i<x;i++)//loop to calculate turn aound time and waiting time of each process

{

p[i].c\_t;

p[i].tat=p[i].c\_t-p[i].a\_t;

p[i].w\_t=p[i].tat-p[i].b\_t;

}//below code is for printing the gantt chart

cout<<endl<<endl<<"----------------------------------------------------GANTT CHART--------------------------------------------------------"<<endl<<endl;

cout<<"process\_id"<<" "<<"arrival\_time"<<" "<<"burst\_time"<<" "<<"completion\_time"<<" "<<"turn\_around\_time"<<" "<<"waiting\_time"<<" "<<"response\_time"<<endl;

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

{

cout<<p[i].p\_id<<" "<<p[i].a\_t<<" "<<p[i].b\_t<<" "<<p[i].c\_t<<" "<<p[i].tat<<" "<<p[i].w\_t<<" "<<p[i].r\_t<<endl;

}

return 0;

cout<<"SCHEDULING DONE";

}

**Round-robin in terms of Operating System and purpose of use:-**

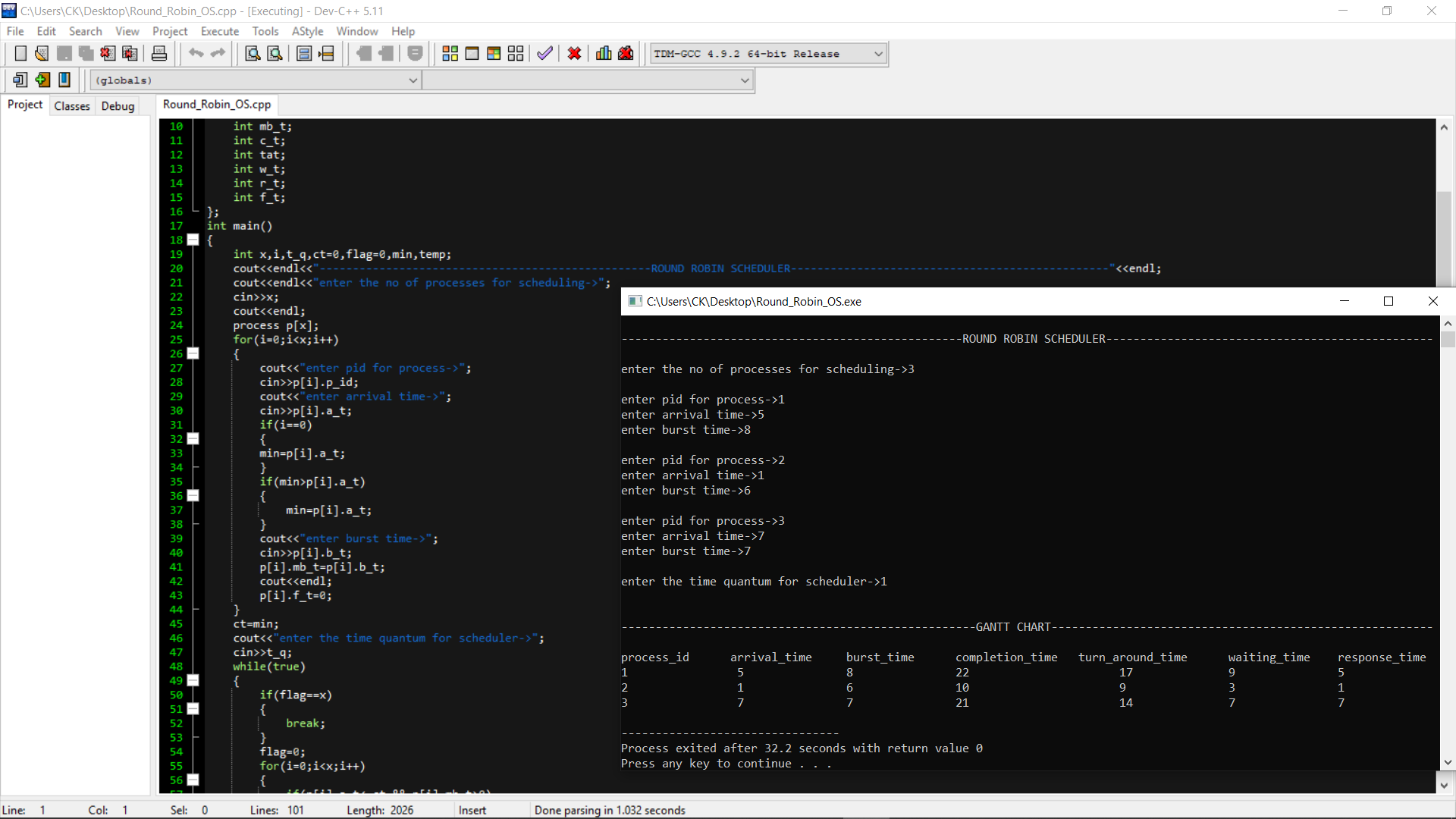
Round-robin (RR) is one of the algorithms employed by process and network schedulers in computing. As the term is generally used, time slices (also known as time quanta) are assigned to each process in equal portions and in circular order, handling all processes without priority(also known as cyclic executive). Round-robin scheduling is simple, easy to implement, and starvation-free. Round-robin scheduling can also be applied to other scheduling problems, such as data packet scheduling in computer networks. It is an operating system concept.

**Test cases: -**

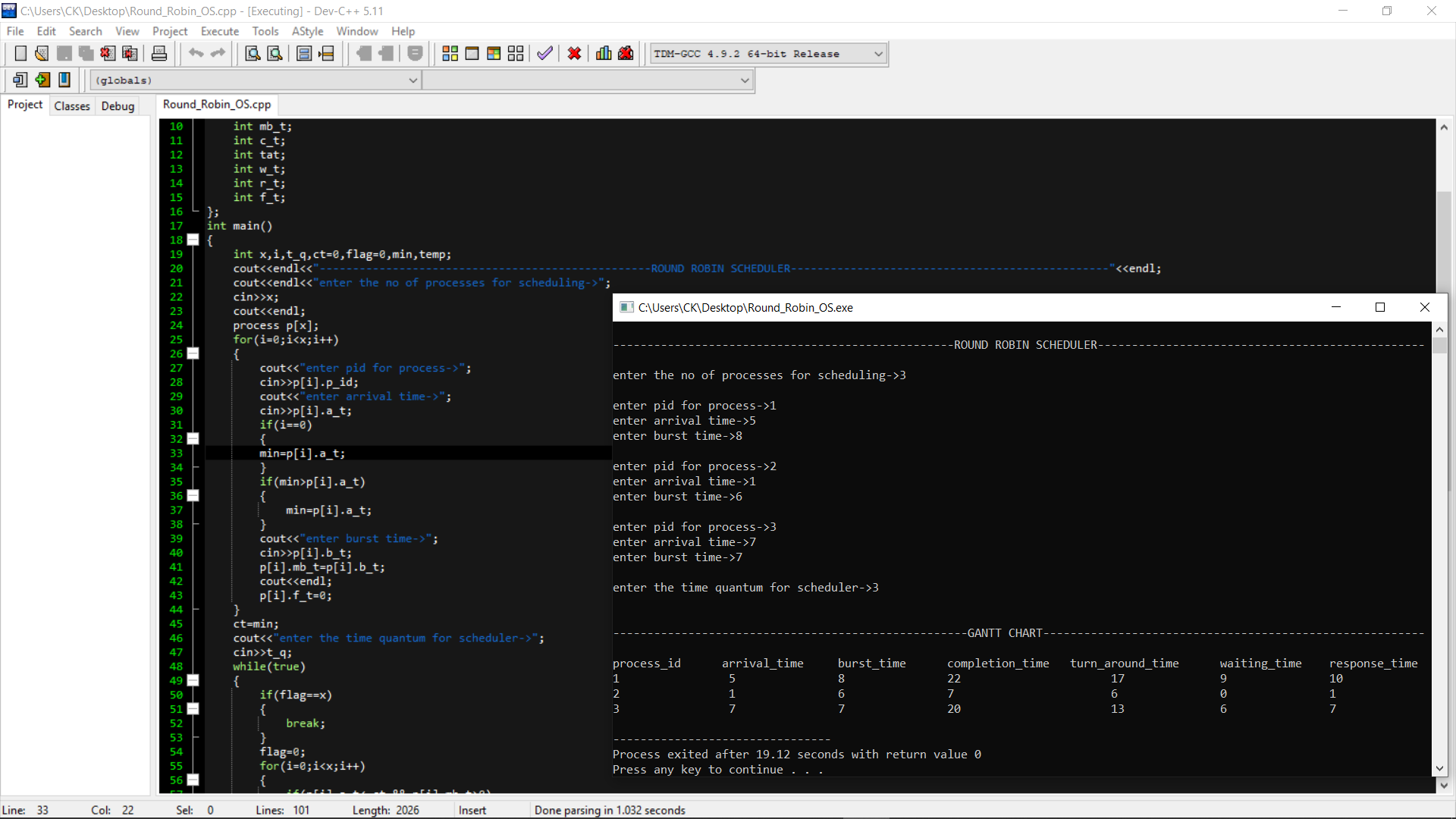
**SAMPLE TEST CASE**

**Number of processes->3 Arrival time->5,1,7 Burst time->8,6,7**

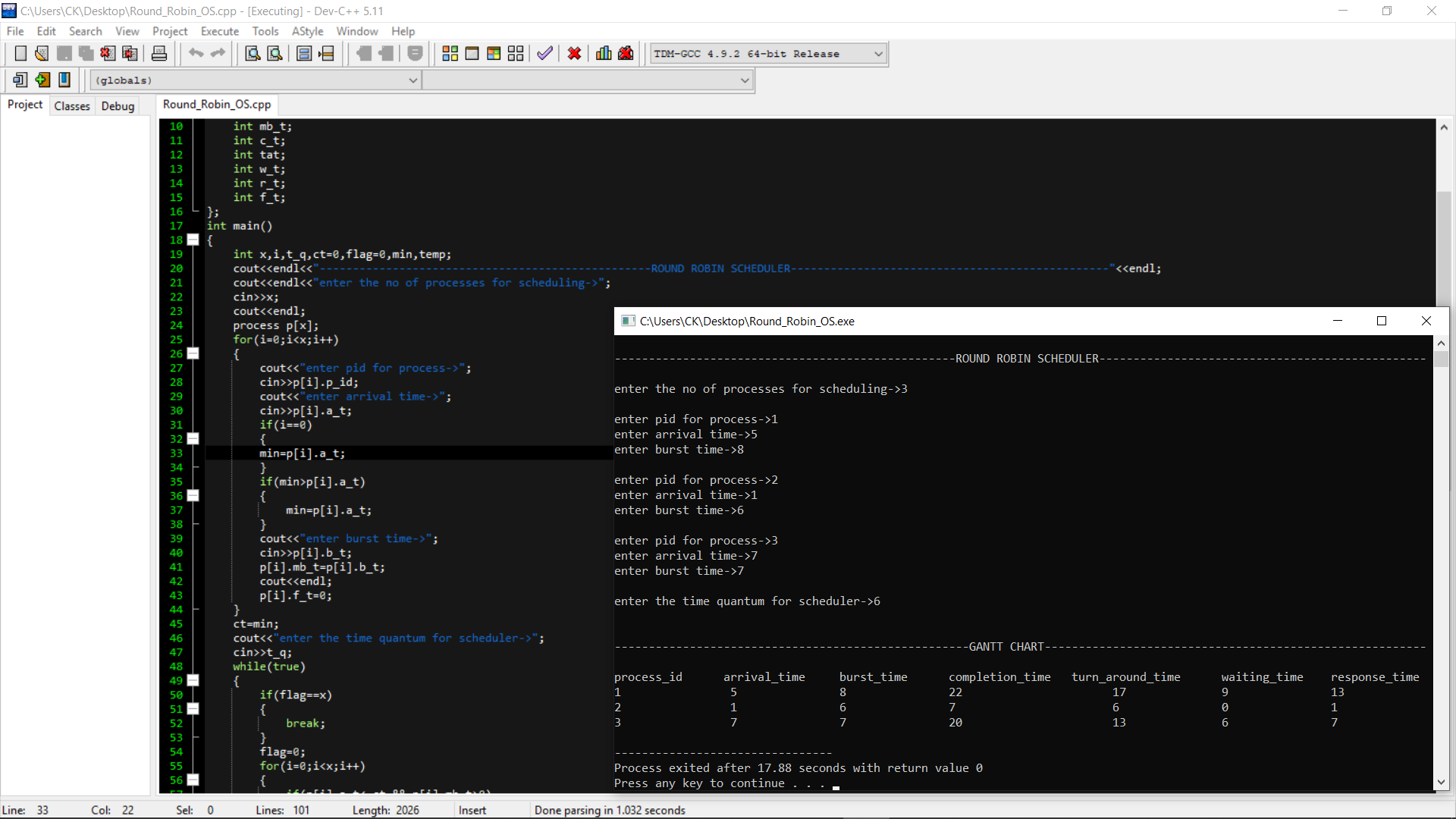
**->Test case with time Quantum=1**

****

**->Test case with time quantum=3**

****

**->Test case with time quantum=6**

****