

CSE316 PROJECT OF OPERATING SYSTEM(OS)

PROGRAMMING ASSESMENT

Submitted by :Rohit Kataria

Reg. ID :11810799

Section and Group: K18SP and Group1

Roll no.: 23

Email add.: mrkataria087@gmail.com

Submitted to : Isha Sharma(23888)

Course: B.Tech(CSE) 2nd year

GITHUB link: <https://github.com/mrkataria>

QUESTION Design a scheduling program to implements a Queue with two levels: Level 1 : Fixed priority preemptive Scheduling Level 2 : Round Robin Scheduling For a Fixed priority preemptive Scheduling (Queue 1), the Priority 0 is highest priority. If one process P1 is scheduled and running, another process P2 with higher priority comes. The New process (high priority) process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considred in the multiples of 2.

All the processes in second level queue will complete their execution according to round robin scheduling. Consider: 1. Queue 2 will be processed after Queue 1 becomes empty. 2. Priority of Queue 2 has lower priority than in Queue 1.

CODE

#include <bits/stdc++.h>

using namespace std;

struct Process\_Data

{

int Num;

int Pid; //Process Id

int A\_time; //Process Arrival Time

int B\_time; //Process Bruest Time

int Priority; //Process Priority

int F\_time; //Process Finish Time

int R\_time; //Process Remaining Time During Execution

int W\_time; //Waiting Time

int S\_time; //Process start Time

int Res\_time;

};

struct Process\_Data current;

typedef struct Process\_Data P\_d ;

bool idsort(const P\_d& x , const P\_d& y)

{

return x.Pid < y.Pid;

}

bool arrivalsort( const P\_d& x ,const P\_d& y)

{

if(x.A\_time < y.A\_time)

return true;

else if(x.A\_time > y.A\_time)

return false;

if(x.Priority < y.Priority)

return true;

else if(x.Priority > y.Priority)

return false;

if(x.Pid < y.Pid)

return true;

return false;

}

bool Numsort( const P\_d& x ,const P\_d& y)

{

return x.Num < y.Num;

}

struct comPare

{

bool operator()(const P\_d& x ,const P\_d& y)

{

if( x.Priority > y.Priority )

return true;

else if( x.Priority < y.Priority )

return false;

if( x.Pid > y.Pid )

return true;

return false;

}

};

void my\_check(vector<P\_d> mv)

{

for(unsigned int i= 0; i < mv.size() ;i++)

{

cout<<" Pid :"<<mv[i].Pid<<" \_time : "<<mv[i].A\_time<<" B\_time : "<<mv[i].B\_time<<" Priority : "<<mv[i].Priority<<endl;

}

}

int main()

{

int i;

vector< P\_d > input;

vector<P\_d> input\_copy;

P\_d temp;

int pq\_process = 0; // for PQ process

int rq\_process = 0; // for RQ process

int A\_time;

int B\_time;

int Pid;

int Priority;

int n;

int clock;

int total\_exection\_time = 0;

cin>>n;

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

{

cin>>Pid>>A\_time>>B\_time>>Priority;

temp.Num = i+1;

temp.A\_time = A\_time;

temp.B\_time = B\_time;

temp.R\_time = B\_time;

temp.Pid = Pid;

temp.Priority = Priority;

input.push\_back(temp);

}

input\_copy = input;

sort( input.begin(), input.end(), arrivalsort );

total\_exection\_time = total\_exection\_time + input[0].A\_time;

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

{

if( total\_exection\_time >= input[i].A\_time )

{

total\_exection\_time = total\_exection\_time +input[i].B\_time;

}

else

{

int diff = (input[i].A\_time - total\_exection\_time);

total\_exection\_time = total\_exection\_time + diff + B\_time;

}

}

int Ghant[total\_exection\_time]={0};

for( i= 0; i< total\_exection\_time; i++ )

{

Ghant[i]=-1;

}

priority\_queue < P\_d ,vector<Process\_Data> ,comPare> pq;

queue< P\_d > rq;

int cpu\_state = 0;

int quantum = 4 ;

current.Pid = -2;

current.Priority = 999999;

for ( clock = 0; clock< total\_exection\_time; clock++ )

{

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

{

if(clock == input[j].A\_time)

{

pq.push(input[j]);

}

}

if(cpu\_state == 0)

{

if(!pq.empty())

{

current = pq.top();

cpu\_state = 1;

pq\_process = 1;

pq.pop();

quantum = 4;

}

else if(!rq.empty())

{

current = rq.front();

cpu\_state = 1;

rq\_process = 1;

rq.pop();

quantum = 4;

}

}

else if(cpu\_state == 1)

{

if(pq\_process == 1 && (!pq.empty()))

{

if(pq.top().Priority < current.Priority )

{

rq.push(current);

current = pq.top();

pq.pop();

quantum = 4;

}

}

else if(rq\_process == 1 && (!pq.empty()))

{

rq.push(current);

current = pq.top();

pq.pop();

rq\_process = 0;

pq\_process = 1;

quantum = 4 ;

}

}

if(current.Pid != -2)

{

current.R\_time--;

quantum--;

Ghant[clock] = current.Pid;

if(current.R\_time == 0)

{

cpu\_state = 0 ;

quantum = 4 ;

current.Pid = -2;

current.Priority = 999999;

rq\_process = 0;

pq\_process = 0;

}

else if(quantum == 0 )

{

rq.push(current);

current.Pid = -2;

current.Priority = 999999;

rq\_process = 0;

pq\_process = 0;

cpu\_state=0;

}

}

}

sort( input.begin(), input.end(), idsort );

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

{

for(int k=total\_exection\_time;k>=0;k--)

{

if(Ghant[k]==i+1)

{

input[i].F\_time=k+1;

break;

}

}

}

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

{

for(int k=0;k<total\_exection\_time;k++)

{

if(Ghant[k]==i+1)

{

input[i].S\_time=k;

break;

}

}

}

sort( input.begin(), input.end(), Numsort );

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

{

input[i].Res\_time=input[i].S\_time-input[i].A\_time;

input[i].W\_time=(input[i].F\_time-input[i].A\_time)-input[i].B\_time;

}

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

{

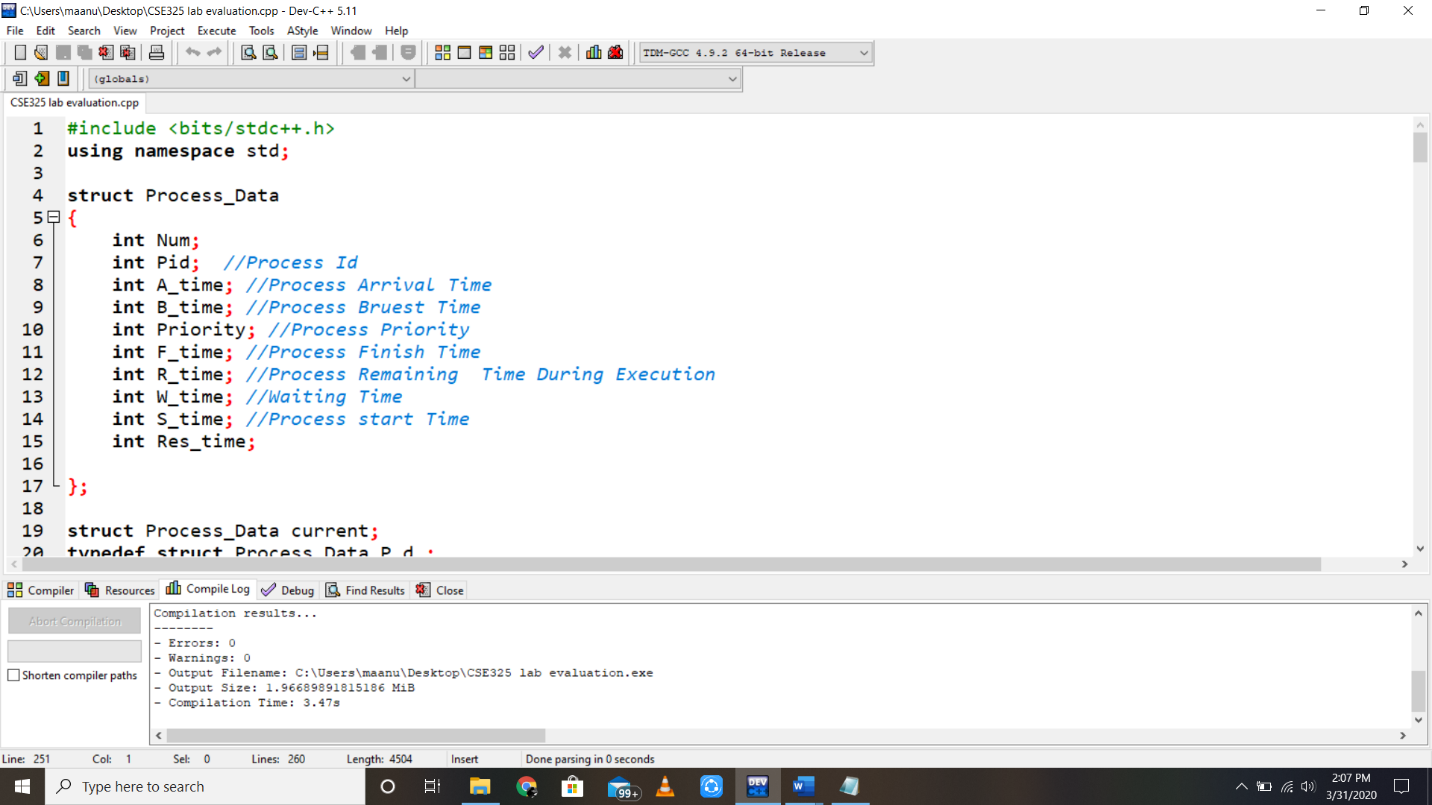
cout<<input[i].Pid<<" "<<input[i].Res\_time<<" "<<input[i].F\_time<<" "<<input[i].W\_time<<endl;

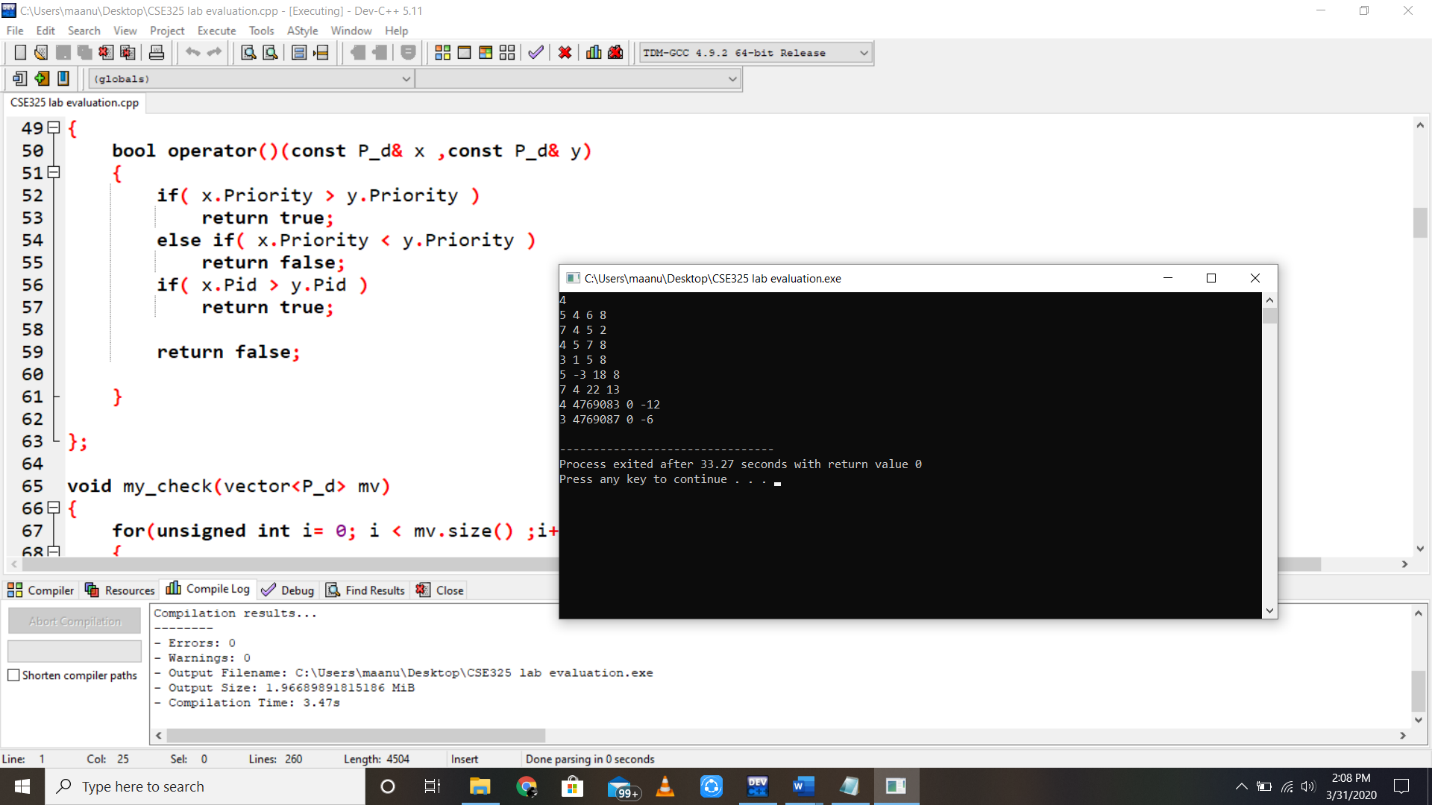
}

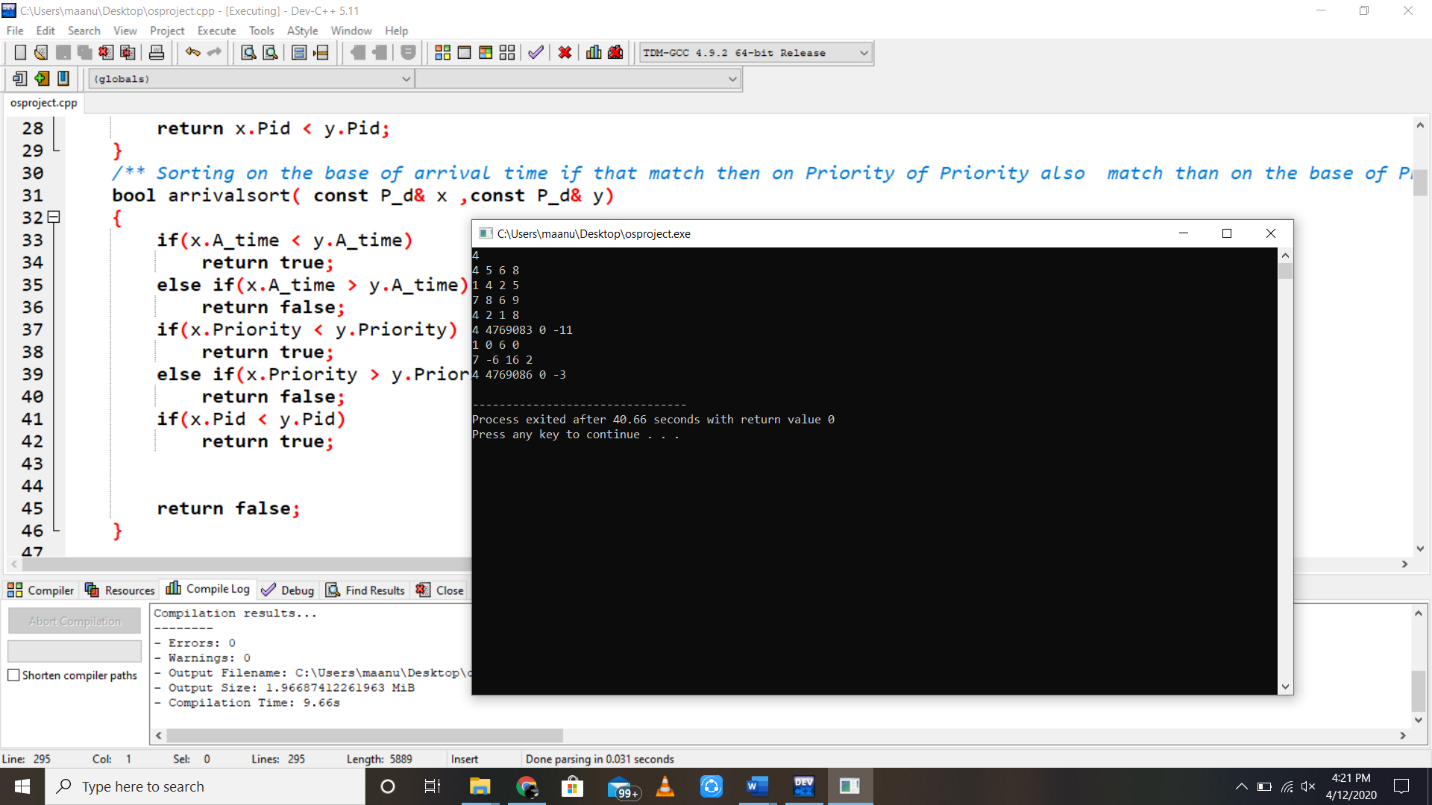
return 0;

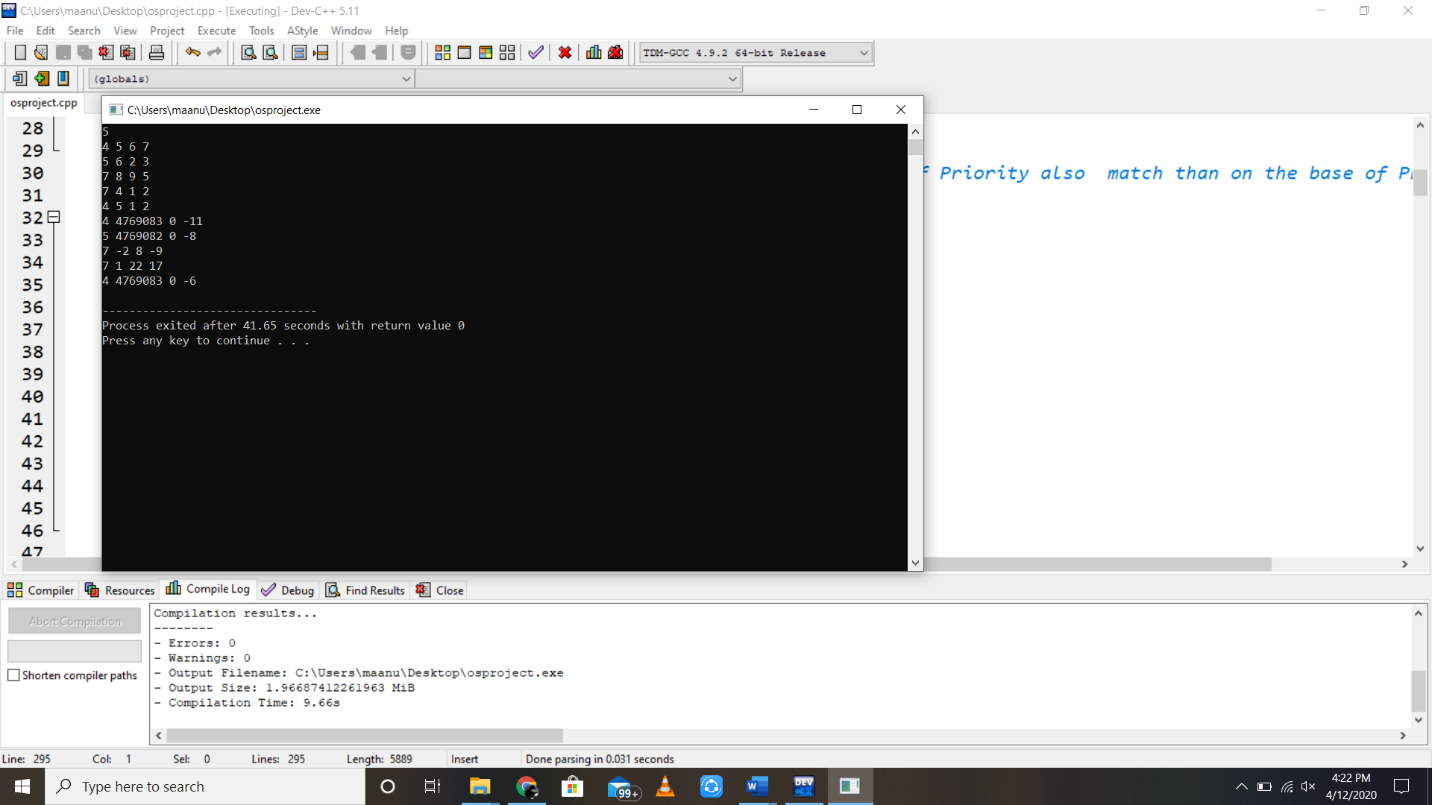
}

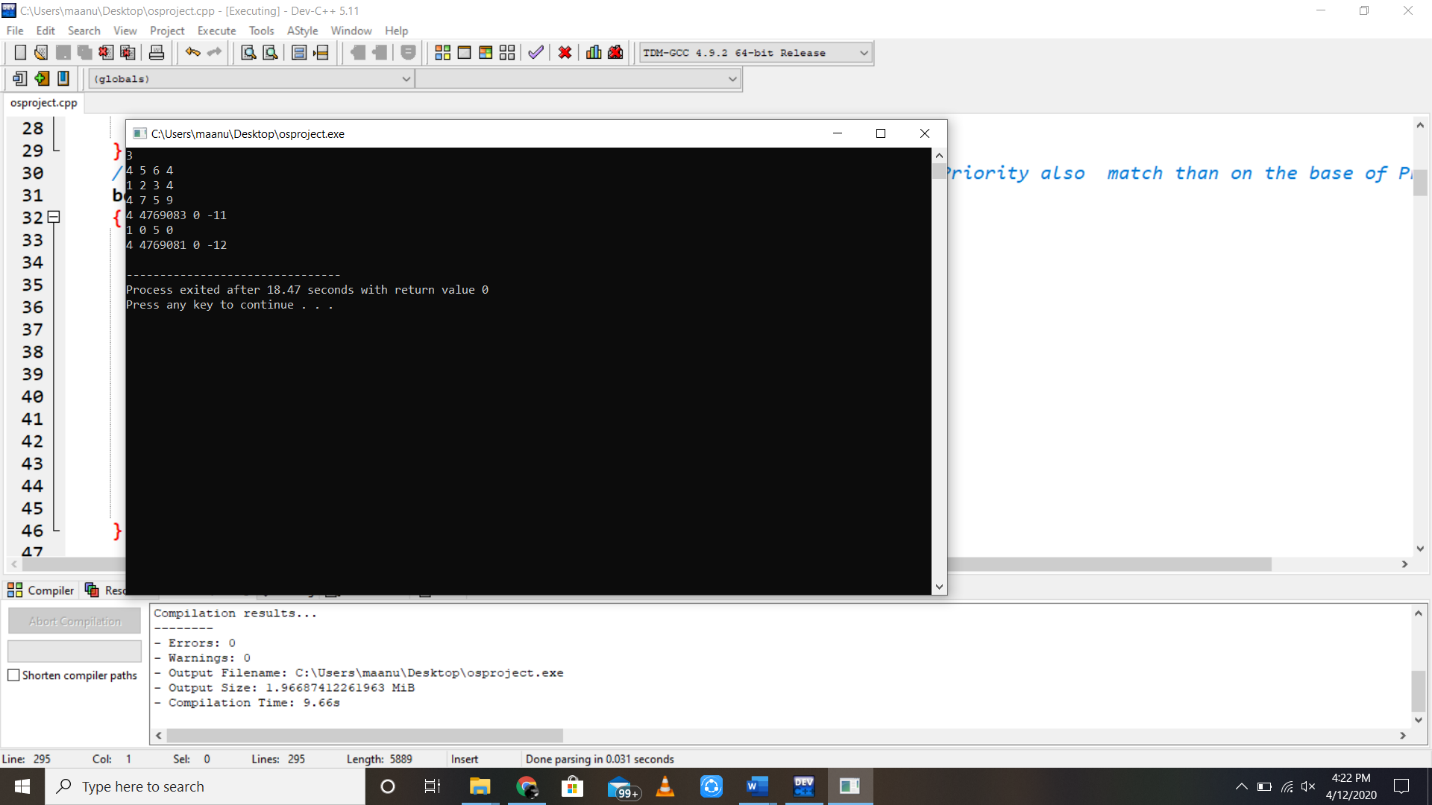
The code is running in the dev c++ compiler:











Screenshots of the running code.

I have taken input element and run it.