LAB TASK – 5

# 1)Linear Queue using arrays:

#include<stdio.h>

#define n 5

int q[n];

int f=-1,r=-1;

void enqueue()

{

int d;

printf("enter the element to insert:");

scanf("%d",&d);

if(r==n-1)

{

printf("overflow");

}

else if(r==-1&&f==-1)

{

f=r=0;

q[r]=d;

}

else

{

r++;

q[r]=d;

}

}

void dequeue()

{

if(f==-1&&r==-1)

{

printf("underflow");

}

else if(f==r)

{

printf(" deleted element is %d\t\n",q[f]);

f=r=-1;

}

else

{

printf(" deleted element is %d\t\n",q[f]);

f++;

}

}

void find()

{

int i,j,k=0;;

printf("enter the element to search:");

scanf("%d",&i);

for( j=f;j<=r;j++)

{

if(q[j]==i)

{

k=1;

break;

}

}

if(k==1)

printf("element is found");

else

printf("element not found");

}

int main()

{

enqueue();

enqueue();

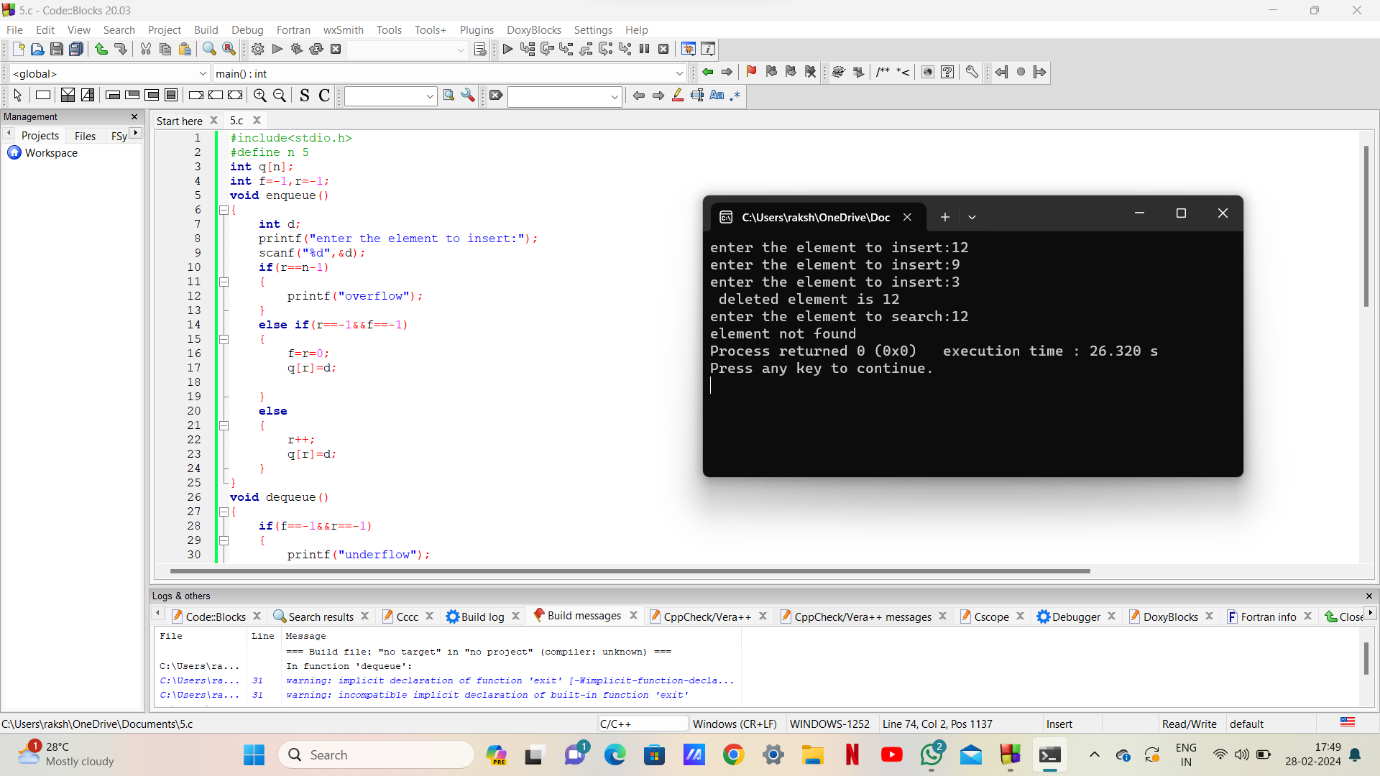
enqueue();

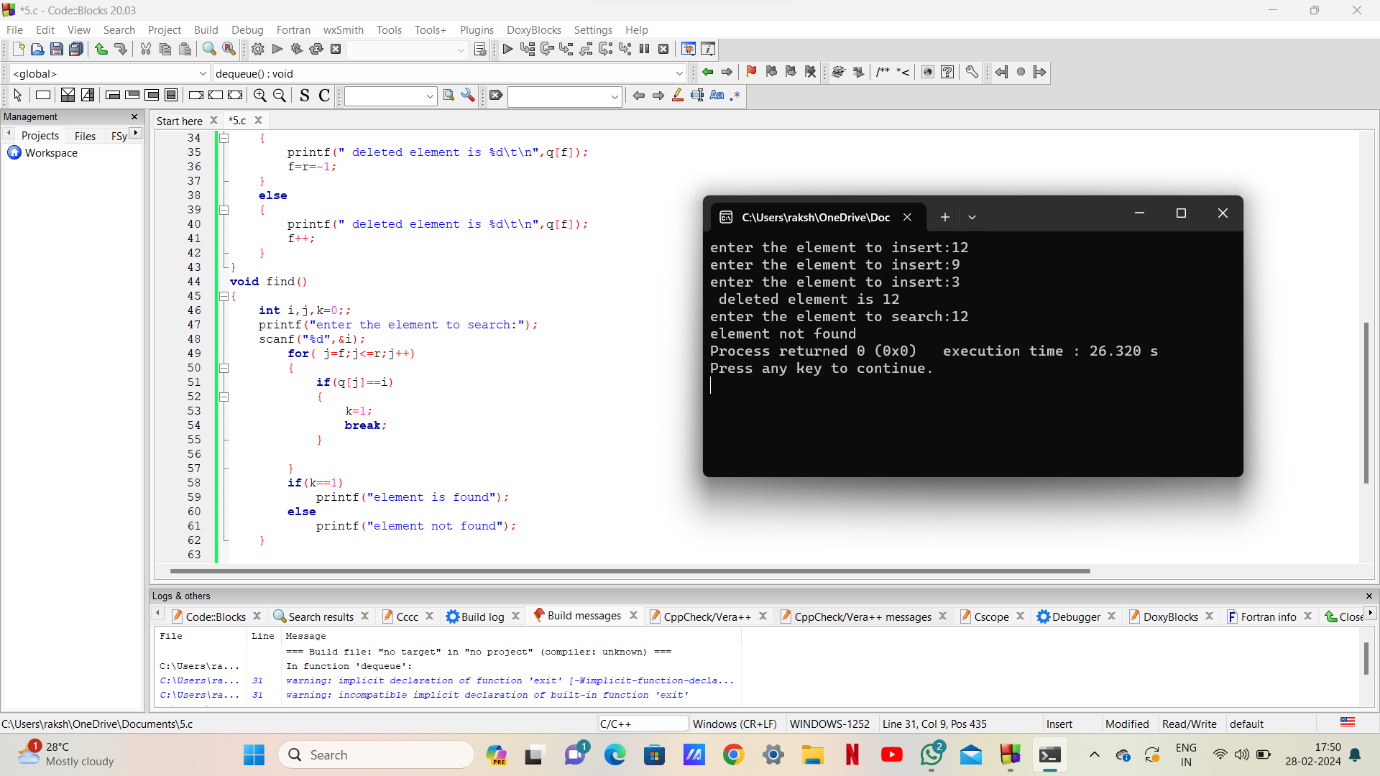
dequeue();

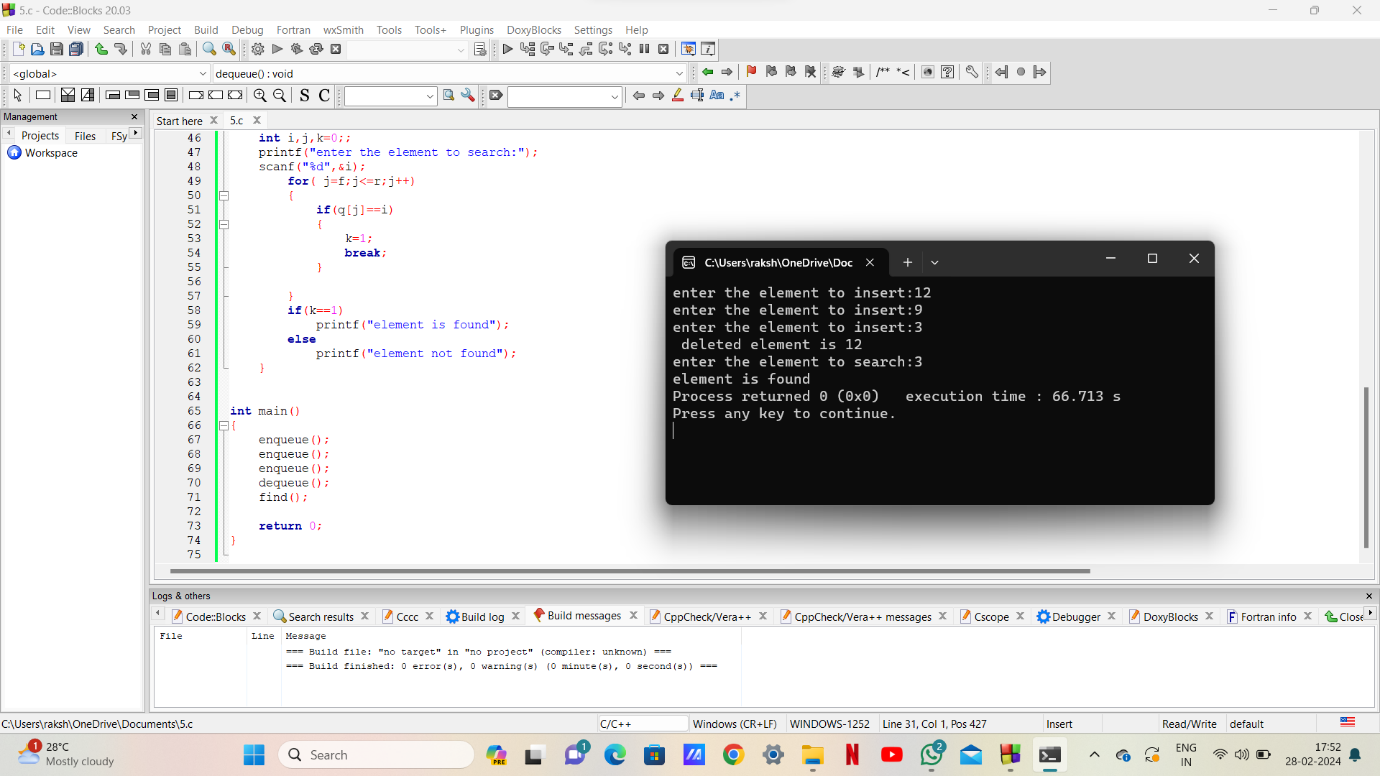
find();

return 0;

}







# 2)Circular Queue using arrays:

#include<stdio.h>

#define n 5

int q[n];

int f=-1,r=-1;

void enqueue()

{

int d;

printf("enter the element to insert:");

scanf("%d",&d);

if(f==-1&r==-1)

{

f=r=0;

q[r]=d;

}

else if(((r+1)%n)==f)

{

printf("\n overflow");

}

else

{

r=(r+1)%n;

q[r]=d;

}

}

void dequeue()

{

if(f==-1&&r==-1)

{

printf("underflow");

}

else if(f==r)

{

printf("deleted element is %d\n",q[f]);

f=r=-1;

}

else

{

printf("deleted element is %d\n",q[f]);

f=(f+1)%n;

}

}

void find()

{

int i,k=0,s;

printf("enter the element to search:");

scanf("%d",&s);

for(i=f;i!=r+1;i=(i+1)%n)

{

if(s==q[i])

{

k=1;

break;

}

}

if(k==1)

printf("element is found");

else

printf("element is not found");

}

int main()

{

enqueue();

enqueue();

enqueue();

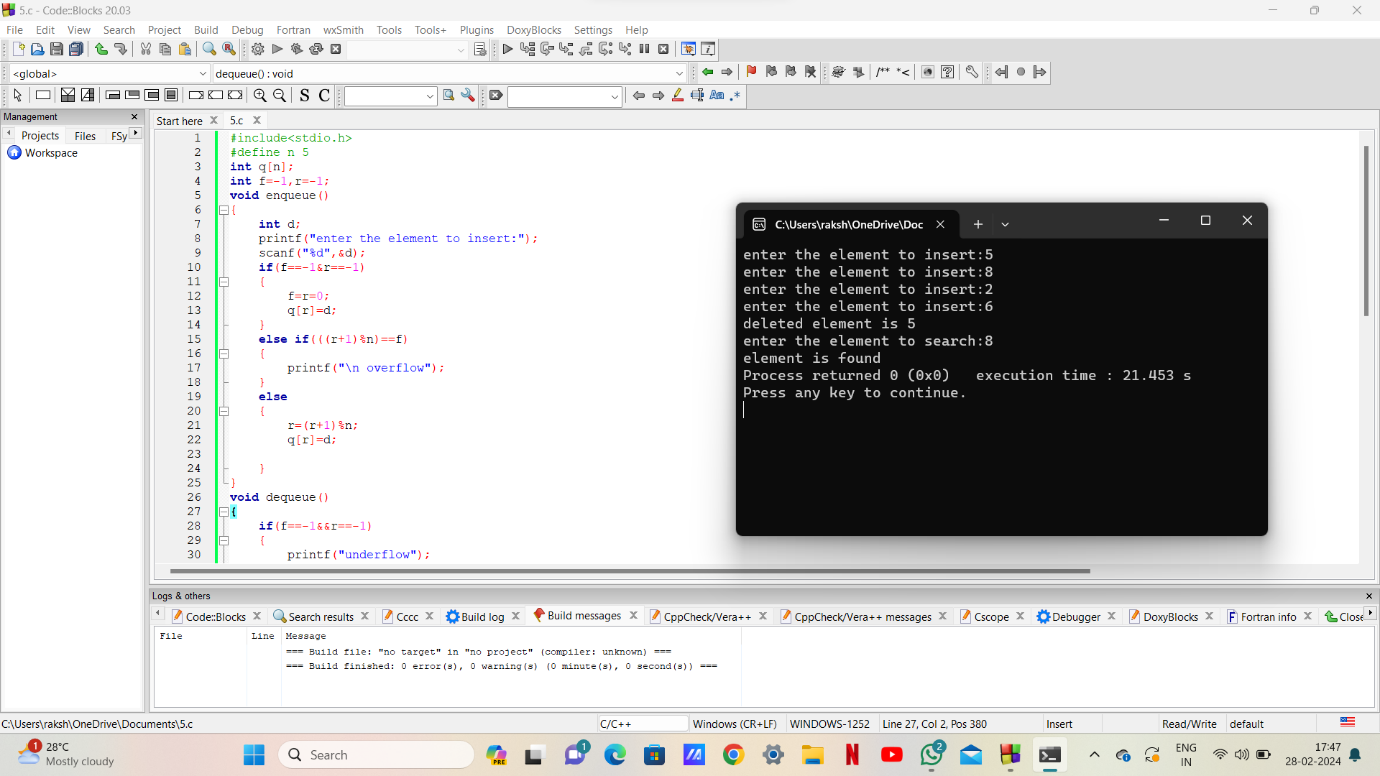
enqueue();

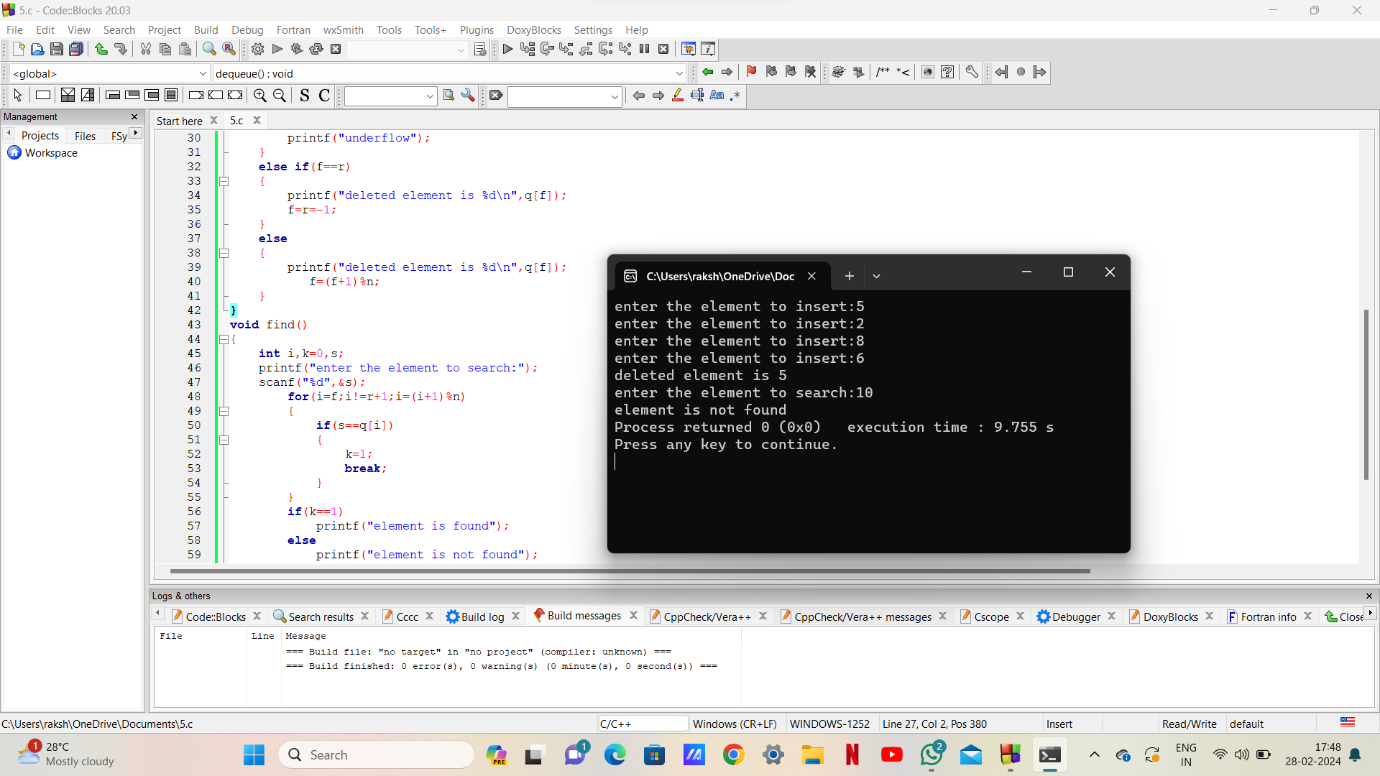
dequeue();

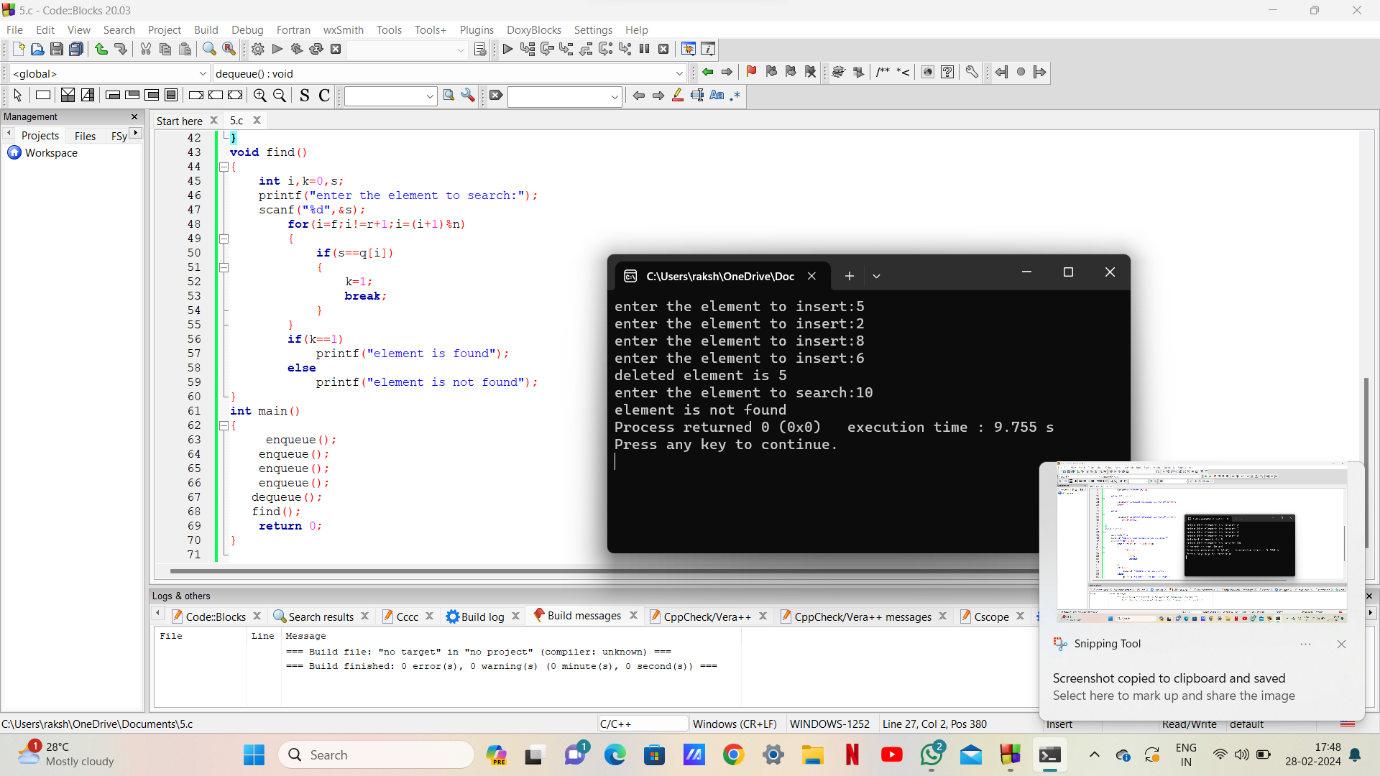
find();

return 0;

}







# 4-Queens Problem:

#include <stdio.h>

#define N 4

int board[N][N];

void printSolution() {

for (int i = 0; i < N; i++) {

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

printf("%d ", board[i][j]);

printf("\n");

}

printf("\n");

}

int isSafe(int row, int col) {

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

if (board[row][i])

return 0;

for (int i = row, j = col; i >= 0 && j >= 0; i--, j--)

if (board[i][j])

return 0;

for (int i = row, j = col; j >= 0 && i < N; i++, j--)

if (board[i][j])

return 0;

return 1;

}

int solveNQueens(int col) {

if (col >= N) {

printSolution();

return 1;

}

int res = 0;

for (int i = 0; i < N; i++) {

if (isSafe(i, col)) {

board[i][col] = 1;

res = solveNQueens(col + 1) || res;

board[i][col] = 0;

}

}

return res;

}

int main() {

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

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

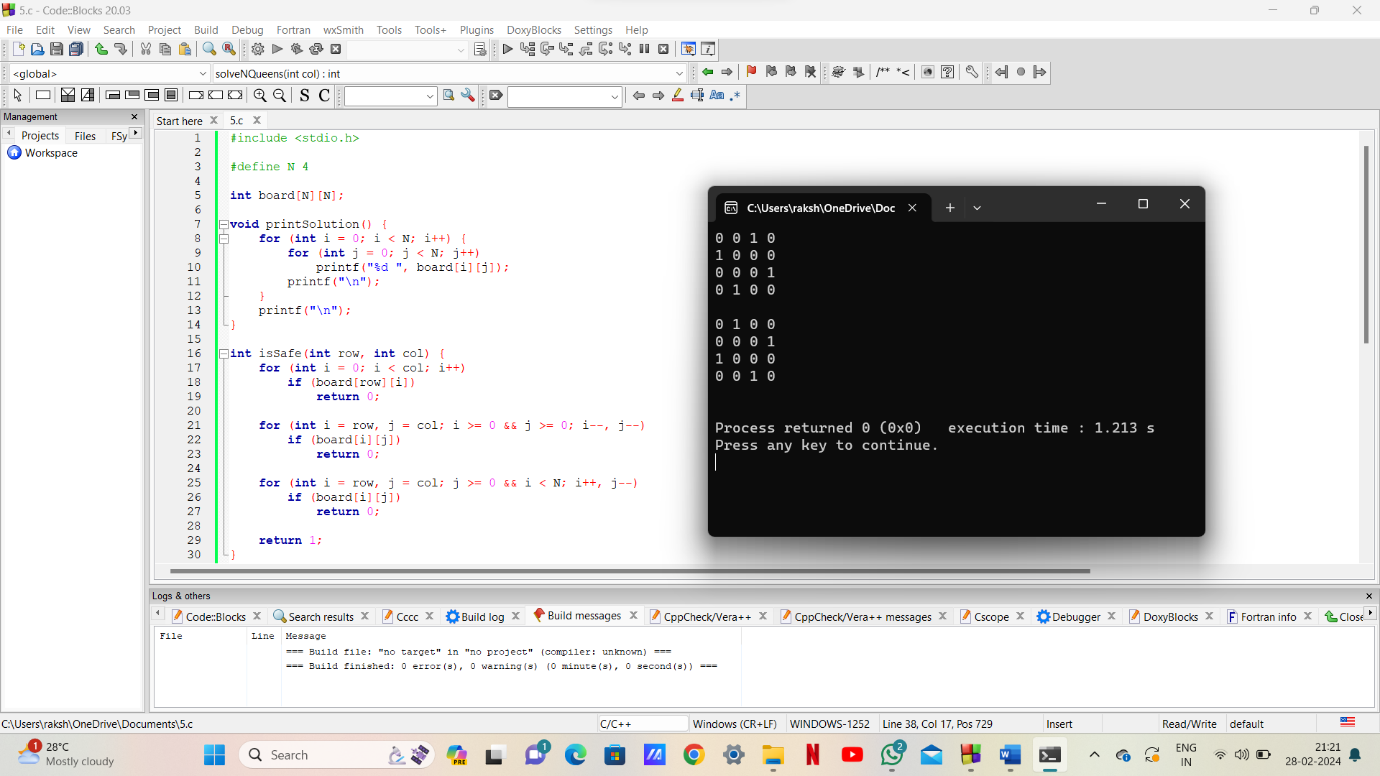
board[i][j] = 0;

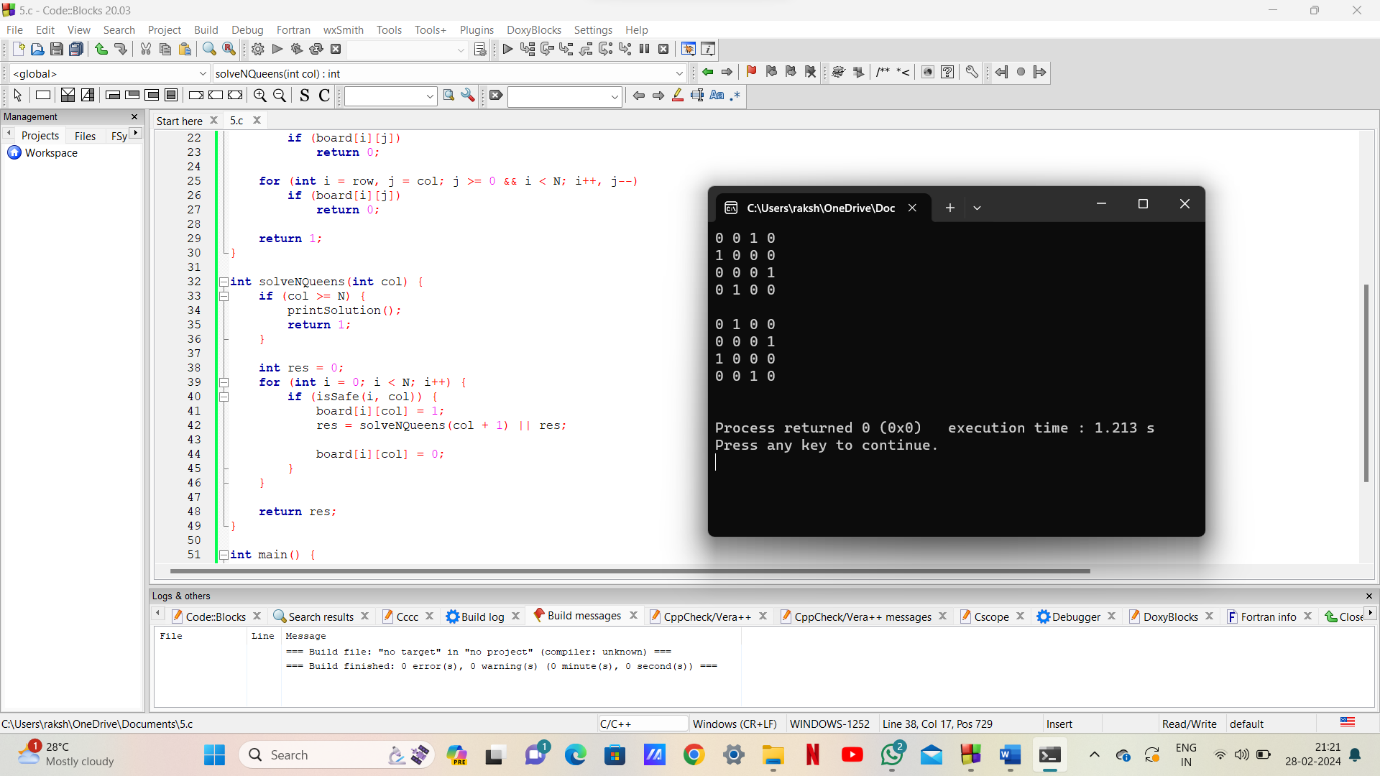
if (!solveNQueens(0))

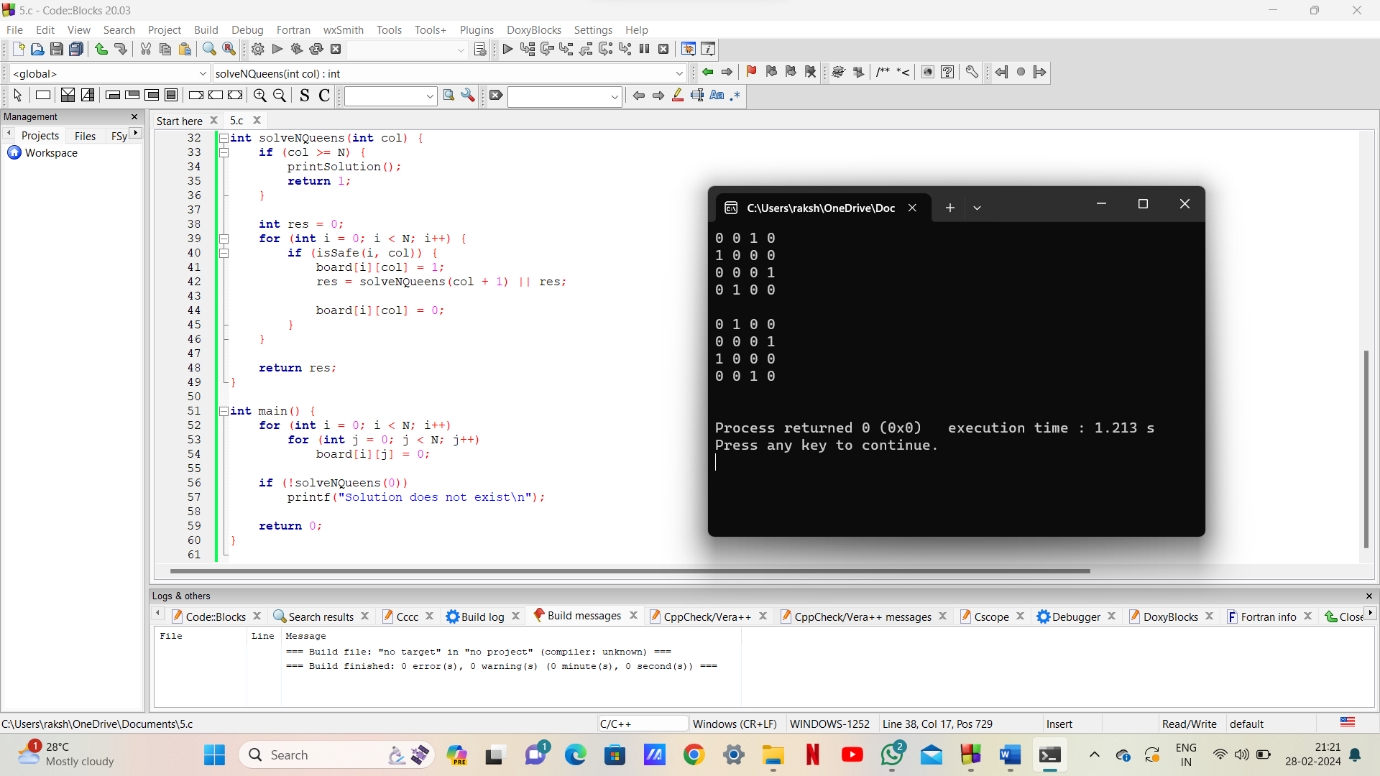
printf("Solution does not exist\n");

return 0;

}







\*\*\*THE END\*\*\*