ASSIGNMENT NO 08

Name : Vishal Sule

Batch : D3

Roll no : 234

PRN No- 0120190064

CODE :

#include<iostream>

using namespace std;

int grid[10][10];

//print the solution

void print(int n)

{

for (int i = 0;i <= n-1; i++) {

for (int j = 0;j <= n-1; j++) {

cout <<grid[i][j]<< " ";

}

cout<<endl;

}

cout<<endl;

cout<<endl;

}

//function for check the position is safe or not

//row is indicates the queen no. and col represents the possible positions

bool isSafe(int col, int row, int n) {

//check for same column

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

if (grid[i][col]) {

return false;

}

}

//check for upper left diagonal

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

if (grid[i][j]) {

return false;

}

}

//check for upper right diagonal

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

if (grid[i][j]) {

return false;

}

}

return true;

}

//function to find the position for each queen

//row is indicates the queen no. and col represents the possible positions

bool solve (int n, int row) {

if (n == row) {

print(n);

return true;

}

//variable res is use for possible backtracking

bool res = false;

for (int i = 0;i <=n-1;i++) {

if (isSafe(i, row, n)) {

grid[row][i] = 1;

//recursive call solve(n, row+1) for next queen (row+1)

res = solve(n, row+1) || res;

//if res ==false then backtracking will occur

//by assigning the grid[row][i] = 0

grid[row][i] = 0;

}

}

return res;

}

int main()

{

ios\_base::sync\_with\_stdio(false);

cin.tie(NULL);

int n;

cout<<"Enter the number of queen"<<endl;

cin >> n;

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

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

grid[i][j] = 0;

}

}

bool res = solve(n, 0);

if(res == false) {

cout << -1 << endl; //if there is no possible solution

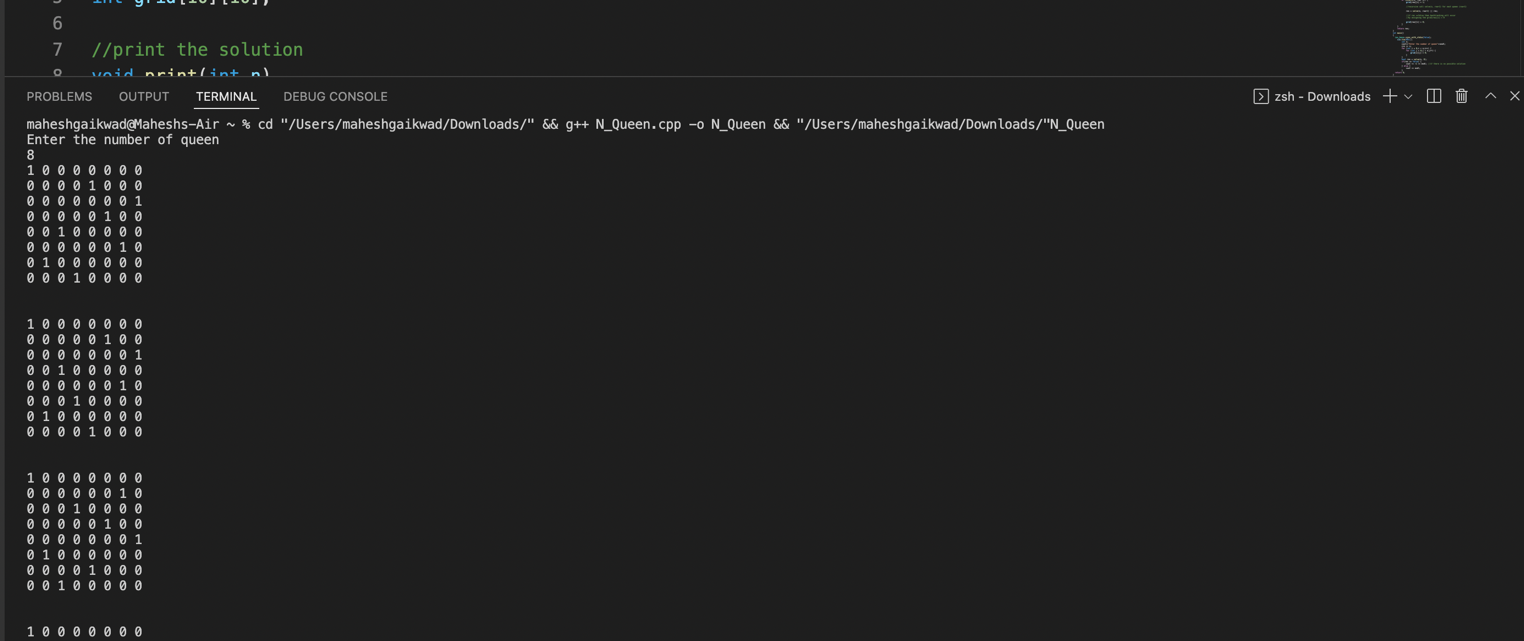
} else {

cout << endl;

}

return 0;

}

OUTPUT :

