Manas Jha
RA1911003010643

AI - Lab 1

Aim: n Queen's Problem

Initial state

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

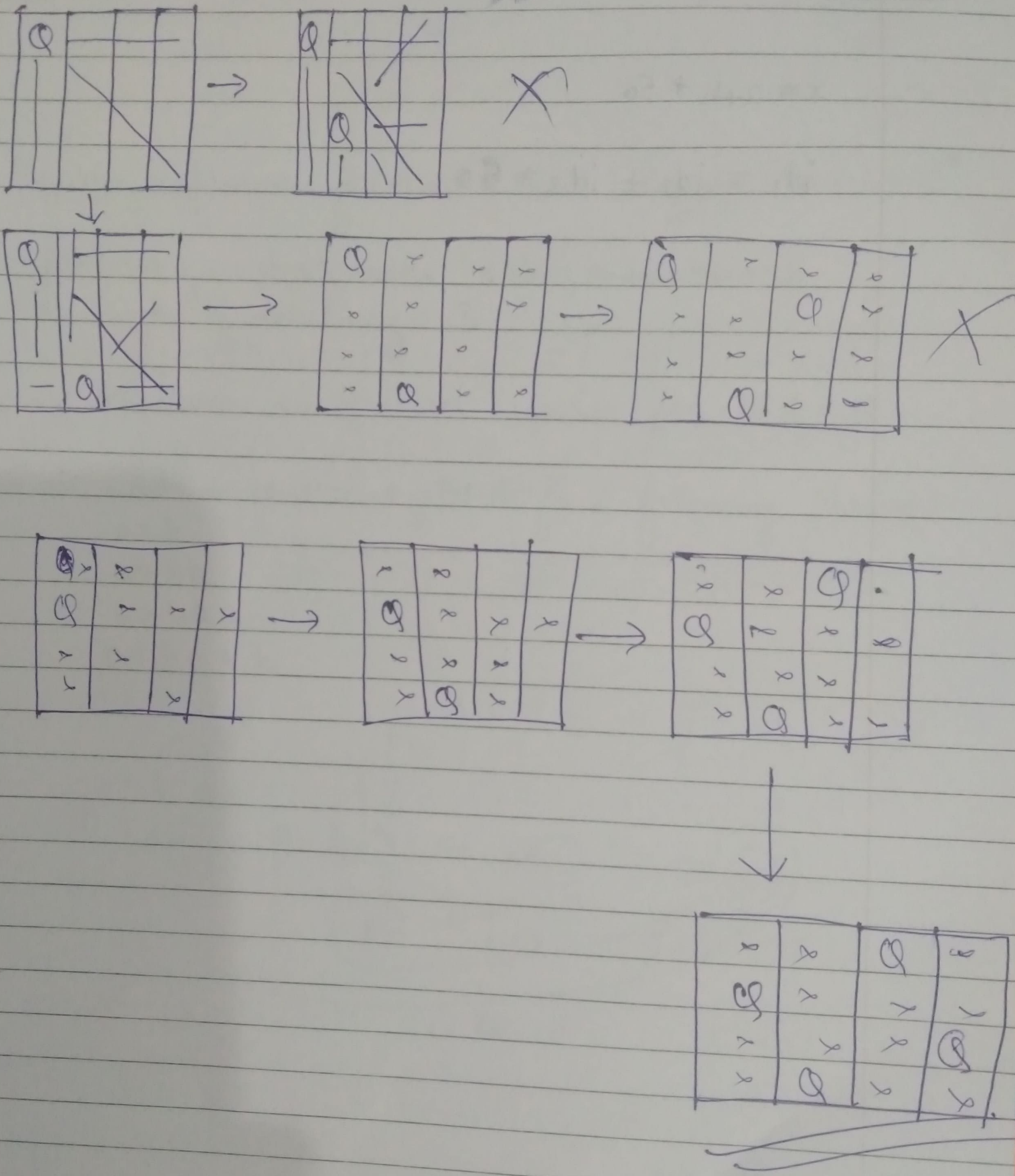
Final state

0	0	1	0
1	0	0	1
0	0	0	0
0	1	0	0

Problem Formulation: To arrange n-queens on a nxn chessboard in such a way that they do not fight each other by being in the same row, column, or diagonal.

Problem Solving:

Date



Manas Jha

RA1911003010643

11/1/2022

n-queen Problem

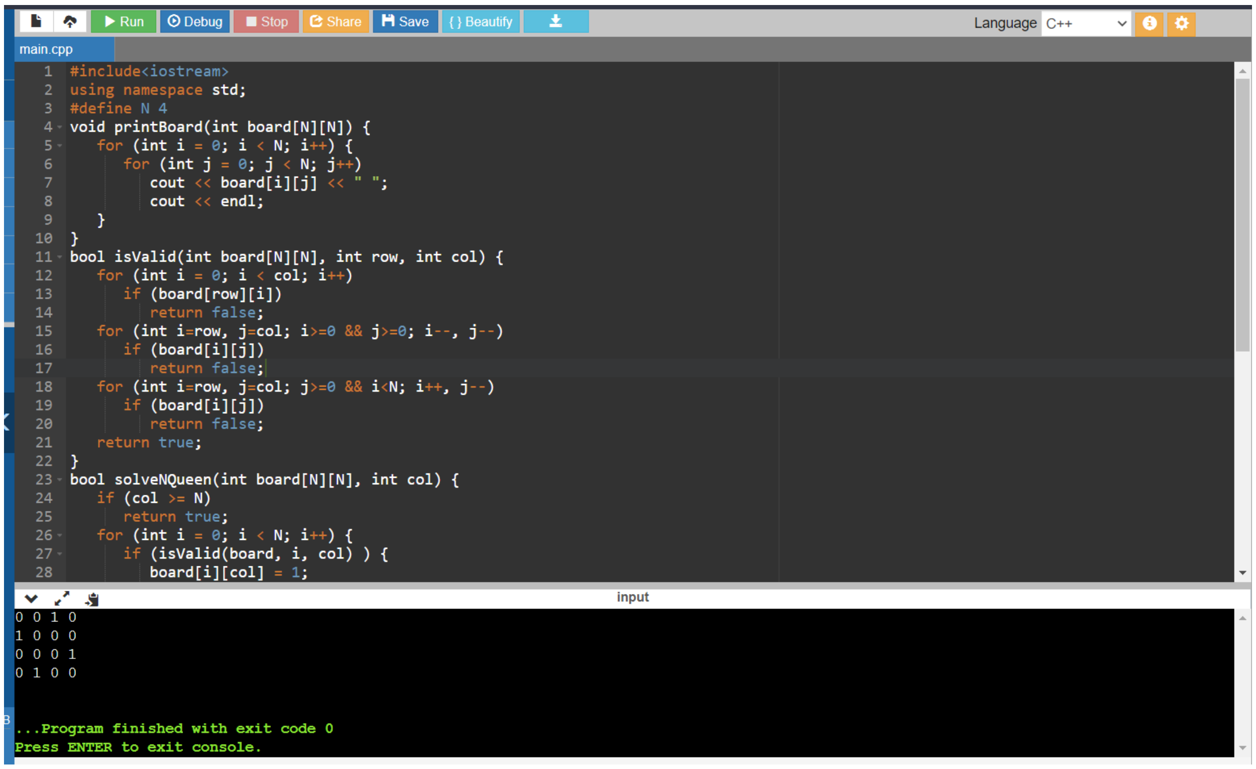
ALGORITHM:

```
If there is a queen at the left of current col, then return false
If there is a queen at the left upper diagonal, then return false
If there is a queen at the left lower diagonal, then return false;
Return true //otherwise it is valid place
If all columns are filled, then return true
For each row of the board, do
If isValid(board, i, col), then
Set queen at place (i, col) in the board
If solveNQueen(board, col+1) = true, then return true
Otherwise remove queen from place (i, col) from board. Done, return false
```

CODE:

```
#include<iostream>
using namespace std;
#define N 4
void printBoard(int board[N][N]) {
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++)
      cout << board[i][j] << " ";
      cout << endl;
  }
}
bool isValid(int board[N][N], int row, int col) {
  for (int i = 0; i < col; i++)
    if (board[row][i])
```

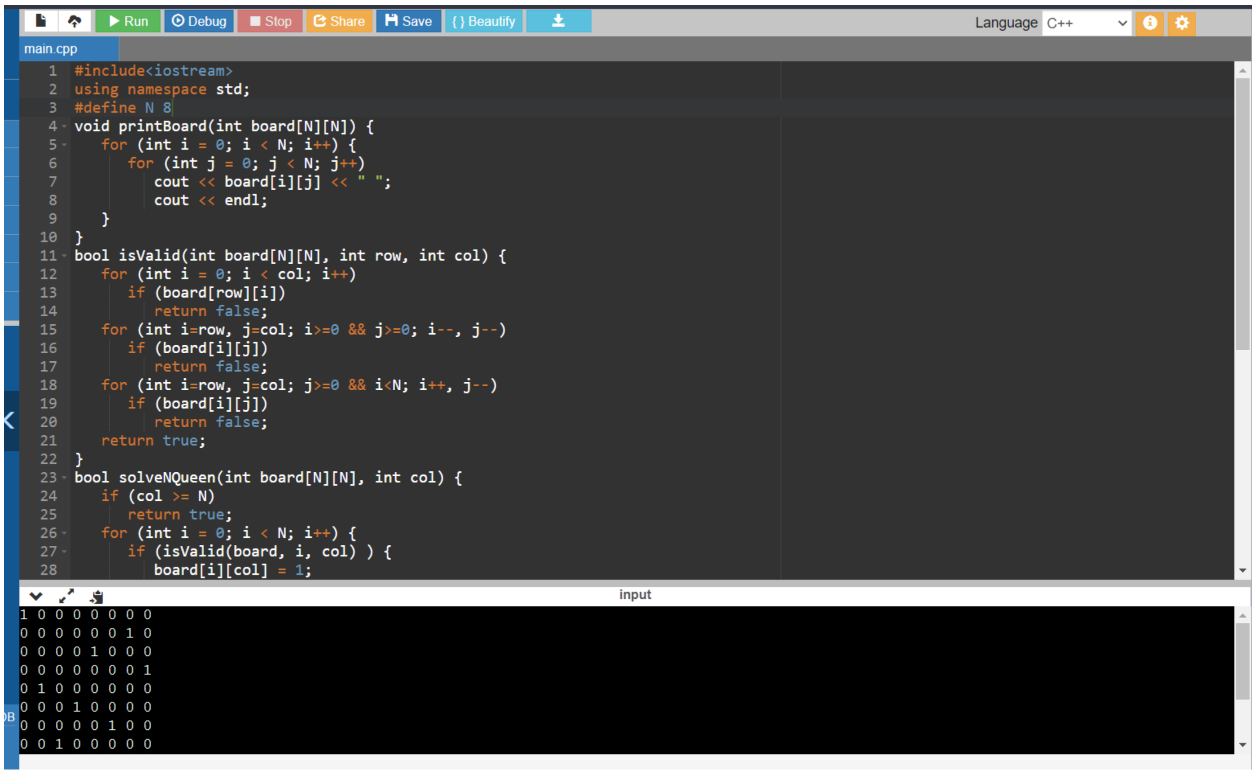
```
            return false;
        for (int i=row, j=col; i>=0 && j>=0; i--, j--)
          if (board[i][j])
            return false;
        for (int i=row, j=col; j>=0 && i<N; i++, j--)
          if (board[i][j])
            return false;
        return true;
      }
      bool solveNQueen(int board[N][N], int col) {
        if (col >= N)
          return true;
        for (int i = 0; i < N; i++) {
          if (isValid(board, i, col) ) {
            board[i][col] = 1;
            if ( solveNQueen(board, col + 1))
              return true;
            board[i][col] = 0;
          }
        }
        return false;
      }
      bool checkSolution() {
        int board[N][N];
        for(int i = 0; i<N; i++)
        for(int j = 0; j<N; j++)
        board[i][j] = 0;
        if ( solveNQueen(board, 0) == false ) {
          cout << "Solution does not exist";
          return false;
        }
        printBoard(board);
        return true;
      }
      int main() {
        checkSolution();
      }
```



```
#include<iostream>
using namespace std;
#define N 4
void printBoard(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << board[i][j] << " ";
            cout << endl;
    }
}
bool isValid(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (int i=row, j=col; i>=0 && j>=0; i--, j--)
        if (board[i][j])
            return false;
    for (int i=row, j=col; j>=0 && i<N; i++, j--)
        if (board[i][j])
            return false;
    return true;
}
bool solveNQueen(int board[N][N], int col) {
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++) {
        if (isValid(board, i, col) ) {
            board[i][col] = 1;
```

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0


...Program finished with exit code 0
Press ENTER to exit console.
```



```cpp
#include<iostream>
using namespace std;
#define N 8
void printBoard(int board[N][N]) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++)
            cout << board[i][j] << " ";
            cout << endl;
    }
}
bool isValid(int board[N][N], int row, int col) {
    for (int i = 0; i < col; i++)
        if (board[row][i])
            return false;
    for (int i=row, j=col; i>=0 && j>=0; i--, j--)
        if (board[i][j])
            return false;
    for (int i=row, j=col; j>=0 && i<N; i++, j--)
        if (board[i][j])
            return false;
    return true;
}
bool solveNQueen(int board[N][N], int col) {
    if (col >= N)
        return true;
    for (int i = 0; i < N; i++) {
        if (isValid(board, i, col) ) {
            board[i][col] = 1;
```

```
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 0 1 0 0
0 0 1 0 0 0 0 0
```