*Muhammad Mufeez*

*BCS-3J*

***DS Lab 05***

**Q6:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

void *displaysolved*(int array[5][5], int N)

{

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

    {

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

        {

            cout *<<* array[i][j] *<<* " ";

        }

        cout *<<* *endl*;

    }

}

bool *isValid*(int x, int y, int maze[5][5], int sol[5][5], int n)

{

*if* (x >= 0 && x < n && y >= 0 && y < n && maze[x][y] == 1 && !sol[x][y])

    {

        sol[x][y] = 1;

*return* true;

    }

*return* false;

}

bool *solve*(int x, int y, int maze[5][5], int sol[5][5], int n, int xarr[], int yarr[])

{

*if* (x == n - 1 && y == n - 1)

    {

        sol[n - 1][n - 1] = 1;

*displaysolved*(sol, n);

*return* true;

    }

*if* (*isValid*(x, y, maze, sol, n))

    {

*for* (int i = 0; i < 5; ++i)

        {

*if* (*solve*(x + xarr[i], y + yarr[i], maze, sol, n, xarr, yarr))

            {

*return* true;

            }

        }

        sol[x][y] = 0;

*return* false;

    }

*return* false;

}

int *main*()

{

    int n = 5;

    int maze[5][5] = {

        {1, 0, 1, 0, 1},

        {1, 1, 1, 1, 1},

        {0, 1, 0, 1, 1},

        {1, 0, 0, 1, 1},

        {0, 0, 0, 0, 1}};

    int sol[5][5];

    int xarr[n] = {1, 0, 0, -1};

    int yarr[n] = {0, -1, 1, 0};

*for* (int i = 0; i < 5; i++)

    {

*for* (int j = 0; j < 5; ++j)

        {

            sol[i][j] = 0;

        }

    }

*displaysolved*(sol, n);

    cout *<<* *endl*;

*if* (maze[0][0] != 1)

    {

        cout *<<* "Not possible";

*return* 0;

    }

*if* (!*solve*(0, 0, maze, sol, n, xarr, yarr))

    {

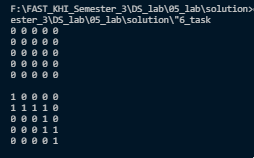
        cout *<<* "Not possible";

    }

*return* 0;

}

*Output:*



**Q7:**

*Code:*

#*include* <bits/stdc++.h>

using namespace std;

void *displaysolved*(char *\*\**array, int N)

{

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

    {

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

        {

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

        }

        cout << endl;

    }

}

void *solve*(int col, char *\*\**board, char *\*\**solved, int N, vector<int> row, vector<int> lowerDiagonal, vector<int> upperDiagonal)

{

*if* (col == N)

    {

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

        {

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

            {

                solved[i][j] = board[i][j];

            }

        }

*return*;

    }

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

    {

*if* (

            row[i] == 0 && lowerDiagonal[i + col] == 0 && upperDiagonal[N - 1 + col - i] == 0)

        {

            board[i][col] = 'Q';

            row[i] = 1;

            lowerDiagonal[i + col] = 1;

            upperDiagonal[N - 1 + col - i] = 1;

*solve*(col + 1, board, solved, N, row, lowerDiagonal, upperDiagonal);

            board[i][col] = '-';

            row[i] = 0;

            lowerDiagonal[i + col] = 0;

            upperDiagonal[N - 1 + col - i] = 0;

        }

    }

}

int *main*()

{

    int N;

    cin >> N;

    char *\*\**board = new char \*[N];

    char *\*\**solved = new char \*[N];

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

    {

        board[i] = new char[N];

        solved[i] = new char[N];

    }

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

    {

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

        {

            board[i][j] = '-';

            solved[i][j] = '-';

        }

    }

    vector<int> *row*(N, 0);

    vector<int> *lowerDiagonal*(N \* 2 - 1, 0);

    vector<int> *upperDiagonal*(N \* 2 - 1, 0);

*solve*(0, board, solved, N, row, lowerDiagonal, upperDiagonal);

*displaysolved*(solved, N);

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

    {

        delete[] board[i];

        delete[] solved[i];

    }

    delete[] board;

    delete[] solved;

*return* 0;

}

*Output:*

