Name: Prathamesh Nandkumar Pandule

Roll No: 307B030

Division: 2

Batch: B

**Assignment No: 3**

**Problem Statement:**

Write a recursive program to find the solution of placing n queens on the chessboard so that no two queens attack each other using Backtracking

**Program:**

#include <iostream>

#define n 5 // Height of Chessboard and number of Queens

using namespace std;

class nqueens

{

public:

// Function to recursively solve the N-Queens problem

void Queen(int board[n][n], int col);

// Function to check if it's safe to place a queen at a specific position

bool place(int board[n][n], int row, int col);

// Function to display the chessboard with queens placed

void display(int board[n][n]);

};

void nqueens::Queen(int board[n][n], int col)

{

if (col >= n)

{

// If all queens are placed, display the solution

display(board);

return;

}

// Try placing queens in each row of the current column

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

{

if (place(board, i, col))

{

// Place a queen at the current position

board[i][col] = 1;

// Recursively solve for the next column

Queen(board, col + 1);

// Backtrack: Remove the queen from the current position

board[i][col] = 0;

}

}

}

bool nqueens::place(int board[n][n], int row, int col)

{

int i, j;

// Check if it's safe to place a queen in the current position

// Check the left side of the current column

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

{

if (board[row][i])

{

return false;

}

}

// Check the upper-left diagonal

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

{

if (board[i][j])

{

return false;

}

}

// Check the lower-left diagonal

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

{

if (board[i][j])

{

return false;

}

}

// If all checks pass, it's safe to place a queen in the current position

return true;

}

void nqueens::display(int board[n][n])

{

// Function to display the chessboard with queens placed

cout << endl

<< "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

cout << "\n";

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

{

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

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

cout << "\n";

}

}

int main(){

nqueens nq;

int board[n][n];

// Initialize the chessboard

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

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

board[i][j] = 0;

// Start solving the N-Queens problem from the first column

nq.Queen(board, 0);

return 0;

}

**Output:**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 0 0 0 0

0 0 0 1 0

0 1 0 0 0

0 0 0 0 1

0 0 1 0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 0 0 0 0

0 0 1 0 0

0 0 0 0 1

0 1 0 0 0

0 0 0 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 1 0 0

1 0 0 0 0

0 0 0 1 0

0 1 0 0 0

0 0 0 0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 0 1 0

1 0 0 0 0

0 0 1 0 0

0 0 0 0 1

0 1 0 0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 1 0 0 0

0 0 0 1 0

1 0 0 0 0

0 0 1 0 0

0 0 0 0 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 0 0 1

0 0 1 0 0

1 0 0 0 0

0 0 0 1 0

0 1 0 0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 1 0 0 0

0 0 0 0 1

0 0 1 0 0

1 0 0 0 0

0 0 0 1 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 0 0 1

0 1 0 0 0

0 0 0 1 0

1 0 0 0 0

0 0 1 0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 0 1 0

0 1 0 0 0

0 0 0 0 1

0 0 1 0 0

1 0 0 0 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

0 0 1 0 0

0 0 0 0 1

0 1 0 0 0

0 0 0 1 0

1 0 0 0 0