**UCID: 2024510001 BATCH: A NAME: Atharva Vasant Angre**

**EXPERIMENT NO: 06**

**EXPERIMENT TITLE:** To implement NQueens problem

5.1 To implement NQueens problem.

5.2 To Understand the backtracking technique.

**Objective:**

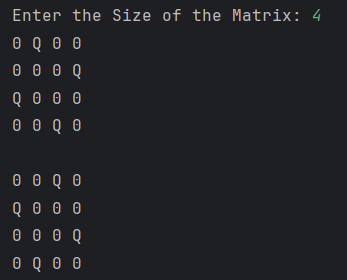
1.To Implement NQueens problem

2.To Understand the backtracking technique

**Program code:** -

import java.util.Scanner;  
  
public class NQueens {  
 public static void main(String[] args) {  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Enter the Size of the Matrix: ");  
 int matrixSize = sc.nextInt();  
 int[][] matrix = new int[matrixSize][matrixSize];  
  
 if (!*makeNQueen*(matrix, 0)) {  
 System.*out*.println("No solution exists.");  
 }  
  
 sc.close();  
 }  
  
 private static boolean makeNQueen(int[][] matrix, int row) {  
 int n = matrix.length;  
  
 if (row >= n) {  
 *printMatrix*(matrix);  
 return true;  
 }  
  
 boolean result = false;  
  
 for (int col = 0; col < n; col++) {  
 if (*isSafeToPlace*(matrix, row, col)) {  
 matrix[row][col] = 1;  
 result = *makeNQueen*(matrix, row + 1) || result;  
 matrix[row][col] = 0;  
 }  
 }  
  
 return result;  
 }  
  
 private static boolean isSafeToPlace(int[][] matrix, int row, int col) {  
 int n = matrix.length;  
  
 for (int i = 0; i < row; i++) {  
 if (matrix[i][col] == 1) return false;  
 }  
  
 for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {  
 if (matrix[i][j] == 1) return false;  
 }  
  
 for (int i = row, j = col; i >= 0 && j < n; i--, j++) {  
 if (matrix[i][j] == 1) return false;  
 }  
  
 return true;  
 }  
  
 private static void printMatrix(int[][] matrix) {  
 for (int[] row : matrix) {  
 for (int cell : row) {  
 System.*out*.print((cell == 1 ? "Q " : "0 "));  
 }  
 System.*out*.println();  
 }  
 System.*out*.println();  
 }  
}

**Output:**

****

**Conclusion:**

In this experiment, we solved the N-Queens problem using backtracking. The program places queens on the board so that they don’t attack each other. If a safe position is found, the queen is placed; otherwise, it backtracks and tries another position. This helped us understand how backtracking works in solving problems step by step. It was a good way to learn how to use recursion for such problems.