Experiment No.10

Name :-Shivani Kolekar

Roll No:-24141031

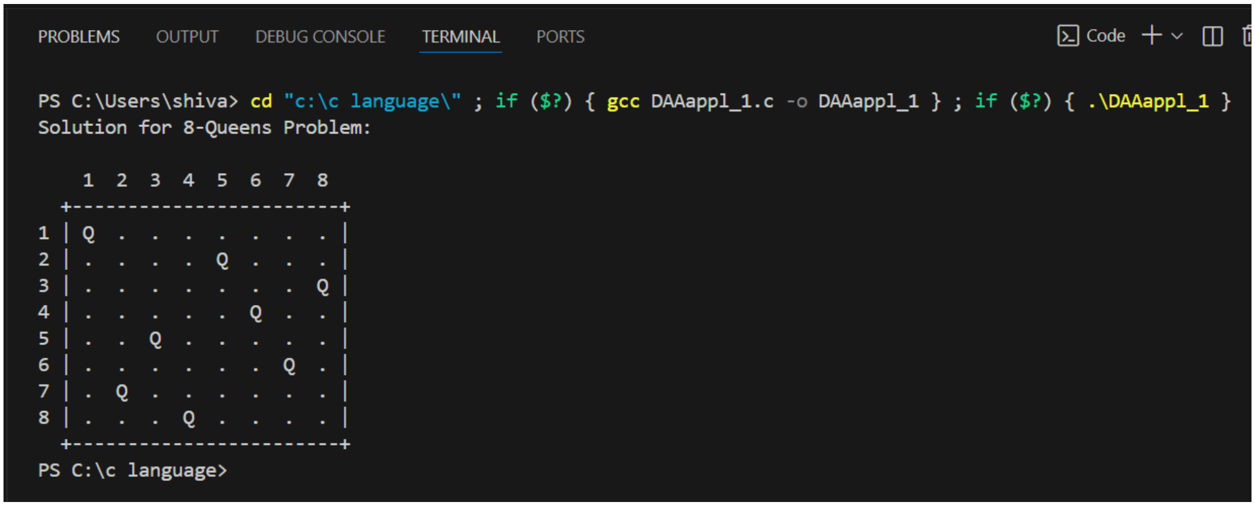
Batch:-I2

Title:-8-Queens Problem using Back Tracking.

Programm:-

```
#include <stdio.h>
#define N 8
char board[N][N];
// Check if it's safe to place a queen at board[row][col]
int isSafe(int row, int col) {
    int i, j;
    // Column check
    for (i = 0; i < row; i++)
        if (board[i][col] == 'Q') return 0;
    // Upper-left diagonal check
    for (i = row, j = col; i >= 0 && j >= 0; i--, j--)
        if (board[i][j] == 'Q') return 0;
    // Upper-right diagonal check
    for (i = row, j = col; i >= 0 && j < N; i--, j++)
        if (board[i][j] == 'Q') return 0;
    return 1;
}
// Recursive function to solve the problem
int solve(int row) {
    if (row >= N) return 1;  // All queens placed
    for (int col = 0; col < N; col++) {
        if (isSafe(row, col)) {
            board[row][col] = 'Q';
            if (solve(row + 1)) return 1;
            board[row][col] = '.';
        }
    }
    return 0;
}
// Function to print the board as a real table
void printBoard() {
    printf("\n    1  2  3  4  5  6  7  8\n");
    printf("  +------------------------+\n");
    for (int i = 0; i < N; i++) {
        printf("%d |", i + 1);
        for (int j = 0; j < N; j++) {
            printf(" %c", board[i][j]);
            printf(" ");
        }
        printf("|\n");
    }
    printf("  +------------------------+\n");
}
int main() {
    // Initialize board
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            board[i][j] = '.';
    if (solve(0)) {
        printf("Solution for 8-Queens Problem:\n");
        printBoard();
    } else {
        printf("No solution exists.\n");
    }
    return 0;
}
```

Output:-



```
PS C:\Users\shiva> cd "c:\c language\" ; if ($?) { gcc DAAappl_1.c -o DAAappl_1 } ; if ($?) { .\DAAappl_1 }
Solution for 8-Queens Problem:

    1  2  3  4  5  6  7  8
  +------------------------+
1 | Q  .  .  .  .  .  .  . |
2 | .  .  .  .  Q  .  .  . |
3 | .  .  .  .  .  .  .  Q |
4 | .  .  .  .  .  Q  .  . |
5 | .  .  Q  .  .  .  .  . |
6 | .  .  .  .  .  .  Q  . |
7 | .  Q  .  .  .  .  .  . |
8 | .  .  .  Q  .  .  .  . |
  +------------------------+
PS C:\c language>
```

Complexity

- **Time Complexity:** O(N!)
- **Space Complexity:** O(N²)

Applications

1. Constraint Satisfaction Problems (CSP)

- 8-Queens is a classic example of CSP in AI, where variables must satisfy constraints.

2. Artificial Intelligence & Game Development
    - Used to teach backtracking algorithms and state-space search techniques.
3. Puzzle Solving & Educational Tools
    - Helps students understand recursion, backtracking, and problem-solving strategies.
4. Optimization Problems
    - Concepts from N-Queens are applied in scheduling, resource allocation, and task assignment.
5. Genetic Algorithms & Heuristics
    - Used as a benchmark problem to test optimization heuristics.
6. Chessboard-related Problems
    - Helps in designing algorithms for chess puzzles and piece placement scenarios.

Conclusion:-The 8-Queens problem demonstrates the power of backtracking to efficiently explore solutions for constraint-based problems.