SUPERIOR UNIVERSITY

Programming For Artificial Intelligence (Lab)

Assignment - 4

Name:

Ali Maqsood.

Roll no:

SU92-BSAIM-F23-050.

Department:

Software Engineering Department.

Program:

Artificial Intelligence.

Section:

BSAI-4A

Question # 1:

Implement the N-Queens Problem (Dynamic)

Code:

```
def print_grid(grid):
    for i in grid:
        print(" ".join("Q" if c else "." for c in i))
    print()


def safe(grid,row,col,n):
    for i in range(row):
        if grid[i][col]:
            return False
    i,j=row-1, col-1
    while i>=0 and j>=0:
        if grid[i][j]:
            return False
        i-=1
        j-=1
    i,j=row-1, col+1
    while i>=0 and j<n:
        if grid[i][j]:
            return False
        i-=1
        j+=1
    return True
```

```
def solve(grid,row,n,outcome):
    if row==n:
        outcome.append([r[:] for r in grid])
        return
    for col in range(n):
        if safe(grid, row, col, n):
            grid[row][col] = 1
            solve(grid, row + 1, n, outcome)
            grid[row][col] = 0

n=int(input("Enter the number of queens: "))
board=[[0]*n for _ in range(n)]
outcome=[]
solve(board,0,n,outcome)

print(f"Total solutions for N={n}: {len(outcome)}\n")
count=1
for sol in outcome:
    print(f"Solution {count}:")
    print_grid(sol)
    count+=1
```

Output:

```
E:\University\4th Semester\2) Programming for Artificial Intellegence (Lab)\Assignments\Assignment 4>python task.py
Enter the number of queens: 4
Total solutions for N=4: 2

Solution 1:
. Q . .
. . . Q
Q . . .
. . Q .

Solution 2:
. . Q .
Q . . .
. . . Q
. Q . .
```

Documentation:

This program solves the N-Queens problem, where we must place N queens on an N-by-N chessboard so that no two queens attack each other. The safe() function checks if placing a queen in a certain spot is safe by looking at the column and diagonals above as well. The solve() function uses recursion to try placing queens row by row to see if a position is safe, it places a queen and moves to the next row; if not, it backtracks and tries a new position. Once all rows are filled safely, that grid is stored in outcome. Finally, the program prints the total number of valid solutions and displays each board using print_grid(), where “Q” represents a queen and “.” an empty space.