**Experiment 18**

**Write a program to implement the dynamic algorithm to solve the Zero-one Knapsack problem.**

**Program:-**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define N 4

void printSolution(int board[N][N]) {

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

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

printf(" %d ", board[i][j]);

printf("\n");

} }

int isSafe(int board[N][N], int row, int col) {

int i, j;

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

if (board[row][i])

return 0;

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

if (board[i][j])

return 0;

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

if (board[i][j])

return 0;

return 1;

}

int solveNQUtil(int board[N][N], int col) {

if (col >= N)

return 1;

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

if (isSafe(board, i, col)) {

board[i][col] = 1;

if (solveNQUtil(board, col + 1))

return 1;

board[i][col] = 0;

} }

return 0;

}

int solveNQ() {

int board[N][N] = { {0, 0, 0, 0},

{0, 0, 0, 0},

{0, 0, 0, 0},

{0, 0, 0, 0} };

if (solveNQUtil(board, 0) == 0) {

printf("Solution does not exist");

return 0;

}

printSolution(board);

return 1;

}

int main() {

double time;

clock\_t start, end;

start = clock();

solveNQ();

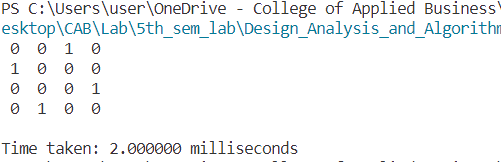
end = clock();

time = ((double)(end - start) \* 1000) / CLOCKS\_PER\_SEC;

printf("\nTime taken: %lf milliseconds\n", time);

return 0;

}**Output:**



**Conclusion:**

This experiment had been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm was implemented in C programming language in Visual Studio Code 1.85.1 Code Editor. The time taken by this algorithm for 4-queen problem is 2 milliseconds.. The running time is analyzed as O(N!) which means there is not the most efficient solution for the N-Queens problem.