**AI LAB ASSIGNMENT 4**

**Name: Atharva Salitri PRN: 12310120**

**Div : CSAI B Batch: 2 Roll No: 37**

**Implement CSP Problems**

**1) Graph Colouring**

**CODE:**

#include <stdio.h>

static int m, n;

static int c = 0;

static int count = 0;

int g[50][50];

int x[50];

void nextValue(int k);

void GraphColoring(int k);

int main()

{

int i, j;

int temp;

printf("\nEnter the number of nodes: ");

scanf("%d", &n);

printf("\nEnter Adjacency Matrix:\n");

for (i = 1; i <= n; i++)

{

for (j = 1; j <= n; j++)

{

scanf("%d", &g[i][j]);

}

}

printf("\nPossible Solutions are: (number = unique colour\n");

for (m = 1; m <= n; m++)

{

if (c == 1)

{

break;

}

GraphColoring(1);

}

printf("\nThe total number of solutions is %d\n", count);

printf("\nThe chromatic number is %d", m - 1);

return 0;

}

void GraphColoring(int k)

{

int i;

while (1)

{

nextValue(k);

if (x[k] == 0)

{

return;

}

if (k == n)

{

c = 1;

for (i = 1; i <= n; i++)

{

printf("%d ", x[i]);

}

count++;

printf("\n");

}

else

{

GraphColoring(k + 1);

}

}

}

void nextValue(int k)

{

int j;

while (1)

{

x[k] = (x[k] + 1) % (m + 1);

if (x[k] == 0)

{

return;

}

for (j = 1; j <= n; j++)

{

if (g[k][j] == 1 && x[k] == x[j])

break;

}

if (j == (n + 1))

{

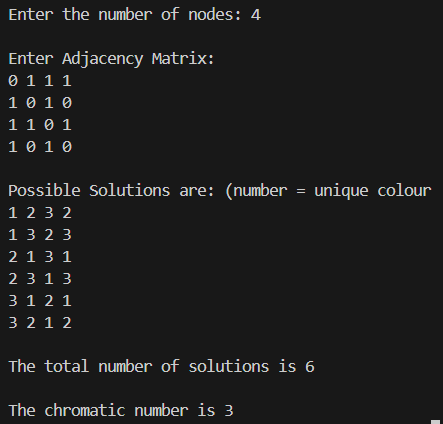
return;

}

}

}

**OUTPUT:**



**2) N-Queens**

**CODE:**

#include <stdio.h>

#include <stdbool.h>

#define MAX 20

int board[MAX][MAX];

int queenPositions[MAX];

void printBoard(int n)

{

printf("Queens positions: [");

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

{

printf("%d", queenPositions[j] + 1);

if (j < n - 1)

printf(", ");

}

printf("]\n");

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

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

{

if (board[i][j] == 1)

printf(" Q ");

else

printf(" . ");

}

printf("\n");

}

printf("\n");

}

bool isSafe(int row, int col, int n)

{

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

{

if (board[row][i])

return false;

}

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

if (board[i][j])

return false;

}

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

if (board[i][j])

return false;

}

return true;

}

void solveNQueens(int col, int n)

{

if (col >= n) {

printBoard(n);

return;

}

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

{

if (isSafe(i, col, n))

{

board[i][col] = 1;

queenPositions[col] = i;

solveNQueens(col + 1, n);

board[i][col] = 0;

}

}

}

int main()

{

int n;

printf("Enter the number of queens (N): ");

scanf("%d", &n);

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

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

board[i][j] = 0;

printf("\nAll possible solutions for %d queens:\n", n);

solveNQueens(0, n);

return 0;

}

**OUTPUT:**

