PRACTICAL 7

Name: Ishita Premchandani
Roll no.: A3_B1_08

Aim: Implement Hamiltonian Cycle using Backtracking.
Problem Statement:
The Smart City Transportation Department is designing a night-patrol route for security vehicles.
Each area of the city is represented as a vertex in a graph, and a road between two areas is represented as an edge.
The goal is to find a route that starts from the main headquarters (Area A), visits each area exactly once, and returns back to the headquarters — forming a Hamiltonian Cycle.
If such a route is not possible, display a suitable message.

1) Adjacency Matrix

	A	B	C	D	E
A	0	1	1	0	1
B	1	0	1	1	0
C	1	1	0	1	0
D	0	1	1	0	1
E	1	0	0	1	0

Code:

```
#include <stdio.h>
#define N 5

int G[N][N] = {
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0},
    {1, 1, 0, 1, 0},
    {0, 1, 1, 0, 1},
    {1, 0, 0, 1, 0}
};

int x[N];

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

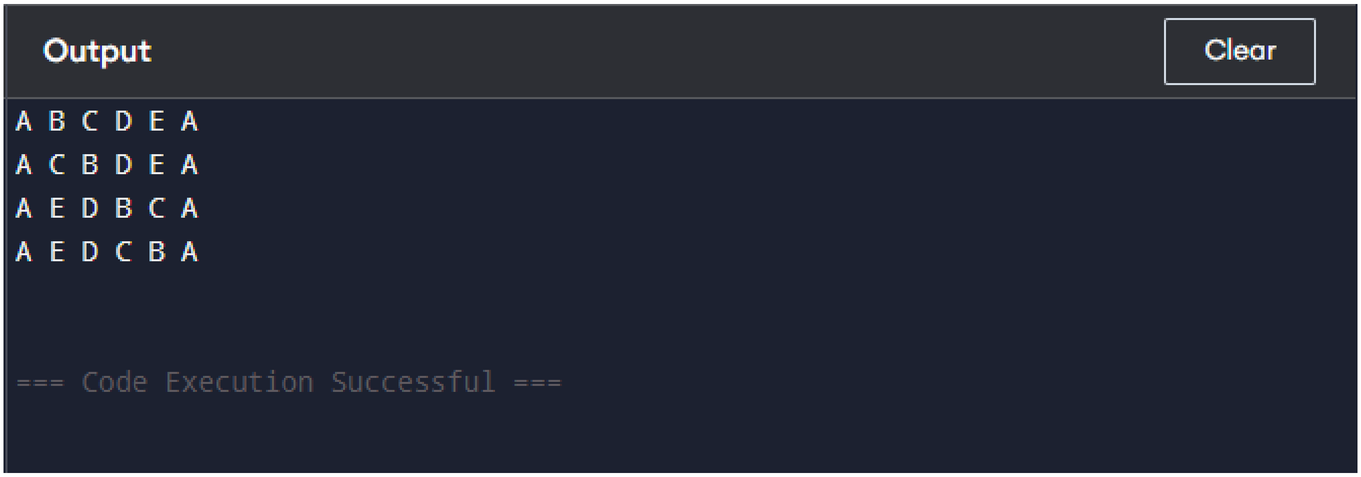
```
        printf("%c ", x[i] + 'A');
    printf("%c\n", x[0] + 'A');
}

void NextVertex(int k) {
    while (1) {
        x[k] = (x[k] + 1) % N;
        if (x[k] == 0)
            return;
        if (G[x[k - 1]][x[k]] != 0) {
            int j;
            for (j = 0; j < k; j++)
                if (x[j] == x[k])
                    break;
            if (j == k) {
                if (k < N - 1 || (k == N - 1 && G[x[k]][x[0]] != 0))
                    return;
            }
        }
    }
}

void Hamiltonian(int k) {
    while (1) {
        NextVertex(k);
        if (x[k] == 0)
            return;
        if (k == N - 1)
            printCycle();
        else
            Hamiltonian(k + 1);
    }
}

int main() {
    for (int i = 0; i < N; i++)
        x[i] = 0;
    x[0] = 0;
    Hamiltonian(1);
    return 0;
}
```

Output:



```
A B C D E A
A C B D E A
A E D B C A
A E D C B A


=== Code Execution Successful ===
```

2) Adjacency Matrix

	T	M	S	H	C
T	0	1	1	0	1
M	1	0	1	1	0
S	1	1	0	1	1
H	0	1	1	0	1
C	1	0	1	1	0

Code:

```
#include <stdio.h>
#define N 5

int G[N][N] = {
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0},
    {1, 1, 0, 1, 1},
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0}
};

int x[N];
```

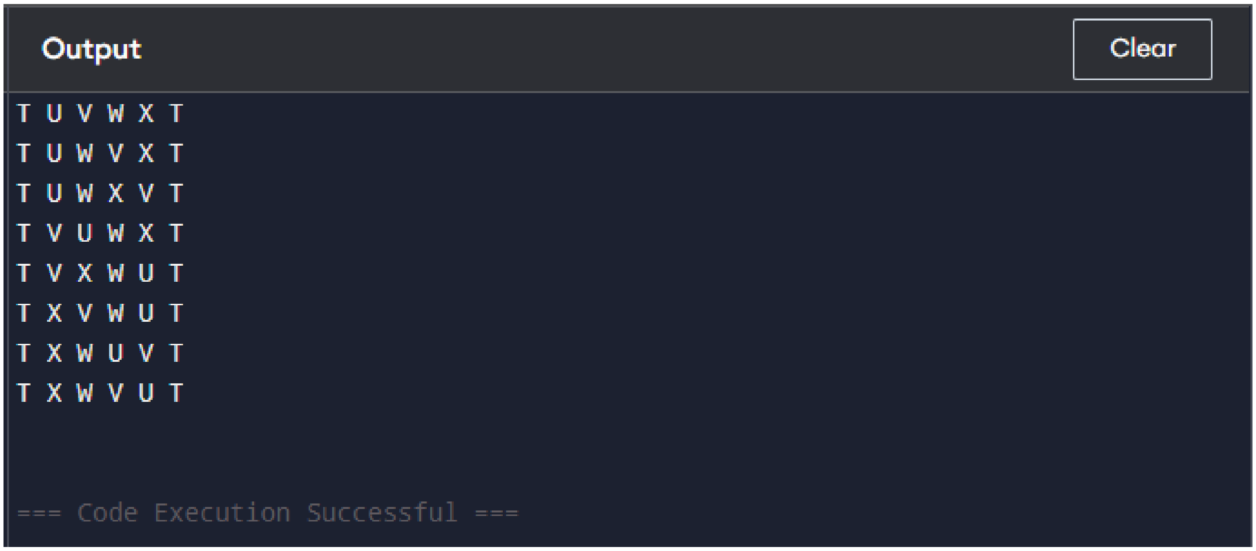
```
void printCycle() {
    for (int i = 0; i < N; i++)
        printf("%c ", x[i] + 'T');
    printf("%c\n", x[0] + 'T');
}

void NextVertex(int k) {
    while (1) {
        x[k] = (x[k] + 1) % N;
        if (x[k] == 0)
            return;
        if (G[x[k - 1]][x[k]] != 0) {
            int j;
            for (j = 0; j < k; j++)
                if (x[j] == x[k])
                    break;
            if (j == k) {
                if (k < N - 1 || (k == N - 1 && G[x[k]][x[0]] != 0))
                    return;
            }
        }
    }
}

void Hamiltonian(int k) {
    while (1) {
        NextVertex(k);
        if (x[k] == 0)
            return;
        if (k == N - 1)
            printCycle();
        else
            Hamiltonian(k + 1);
    }
}

int main() {
    for (int i = 0; i < N; i++)
        x[i] = 0;
    x[0] = 0;
    Hamiltonian(1);
    return 0;
}
```

Output:



```
T U V W X T
T U W V X T
T U W X V T
T V U W X T
T V X W U T
T X V W U T
T X W U V T
T X W V U T


=== Code Execution Successful ===
```