PRACTICAL NO : 07

Name : Yashodeep R Dube

Roll No : A8-B1-1

Aim: Implement Hamiltonian Cycle using Backtracking.

Problem Statement:

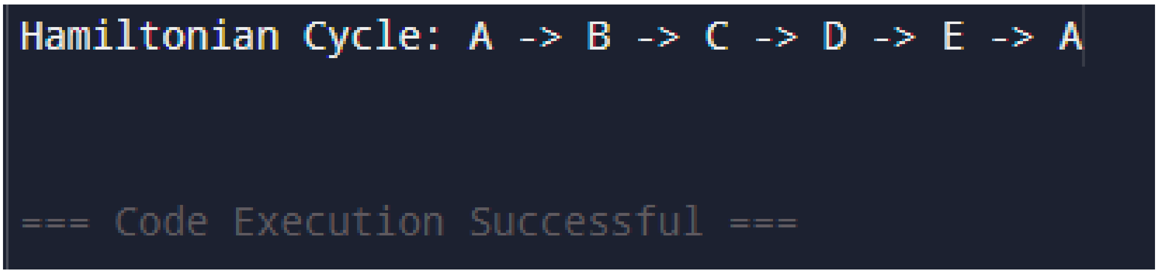
The Smart City Transportation Department is designing a night-patrol route for security vehicles. Each area of the city is represented as a vertex in a graph, and a road between two areas is represented as an edge. The goal is to find a route that starts from the main headquarters (Area A), visits each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

Code :

```
#include <stdio.h>
#define V 5
int graph[V][V] = {
   {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 path[V];
int isSafe(int v, int graph[V][V], int path[], int pos) {
   if (graph[path[pos - 1]][v] == 0)
      return 0;
   for (int i = 0; i < pos; i++)
      if (path[i] == v)
         return 0;
   return 1;
}

int hamCycleUtil(int graph[V][V], int path[], int pos) {
   if (pos == V) {
        if (graph[path[pos - 1]][path[0]] == 1)
            return 1;
        else
            return 0;
    }
    for (int v = 1; v < V; v++) {
        if (isSafe(v, graph, path, pos)) {
            path[pos] = v;
            if (hamCycleUtil(graph, path, pos + 1))
                return 1;
            path[pos] = -1;
        }
    }
    return 0;
}
int hamCycle(int graph[V][V]) {
    for (int i = 0; i < V; i++)
        path[i] = -1;
    path[0] = 0;
    if (!hamCycleUtil(graph, path, 1)) {
        printf("No Hamiltonian Cycle\n");
        return 0;
    }
    printf("Hamiltonian Cycle: ");
    for (int i = 0; i < V; i++)
        printf("%c -> ", 'A' + path[i]);
    printf("%c\n", 'A' + path[0]);
    return 1;
}
int main() {
    hamCycle(graph);
    return 0;
}
```

Output :



```
Hamiltonian Cycle: A -> B -> C -> D -> E -> A


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