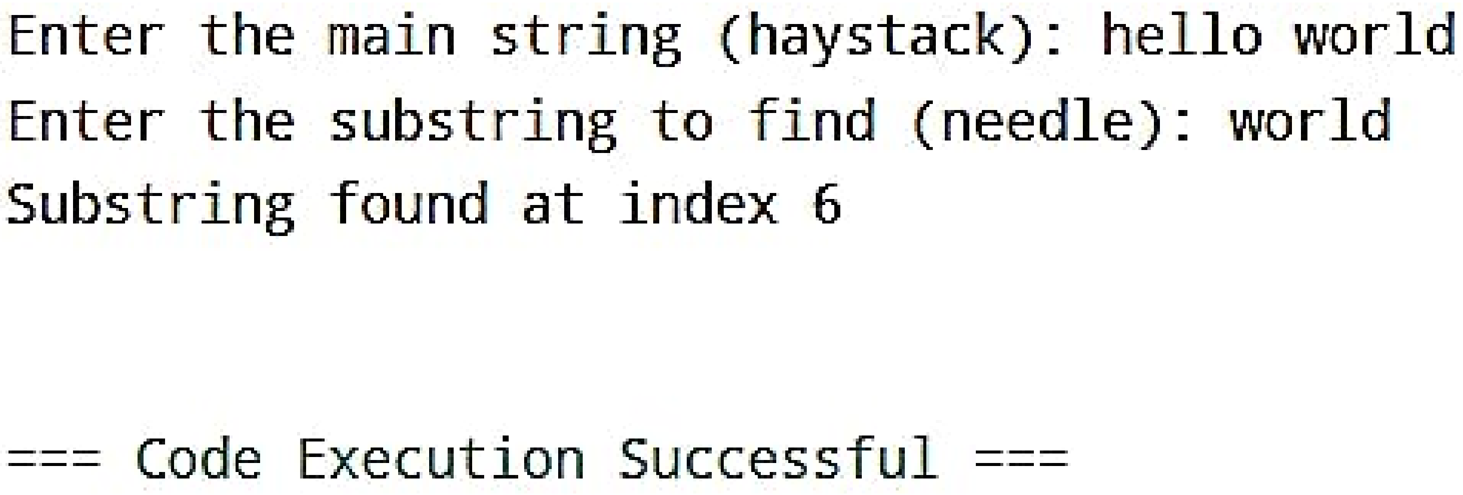
```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 50

// Structure to represent an edge
struct Edge {
    int src, dest, weight;
};

// Structure for Disjoint Set (Union-Find)
int parent[MAX];

int find(int i) {
    if (parent[i] == i)
        return i;
    return parent[i] = find(parent[i]);
}

void unionSet(int u, int v) {
    int set_u = find(u);
    int set_v = find(v);
    parent[set_u] = set_v;
}

int main() {
    int n;
    int graph[MAX][MAX];
```

```c
struct Edge edges[MAX * MAX];
int edgeCount = 0;

printf("Enter number of vertices: ");
scanf("%d", &n);

printf("Enter adjacency matrix (0 if no edge):\n");
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        scanf("%d", &graph[i][j]);
    }
}

// Convert adjacency matrix to edge list
for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < n; j++) {
        if (graph[i][j] != 0) {
            edges[edgeCount].src = i;
            edges[edgeCount].dest = j;
            edges[edgeCount].weight = graph[i][j];
            edgeCount++;
        }
    }
}

// Sort edges by weight (simple bubble sort for clarity)
for (int i = 0; i < edgeCount - 1; i++) {
    for (int j = 0; j < edgeCount - i - 1; j++) {
        if (edges[j].weight > edges[j + 1].weight) {
            struct Edge temp = edges[j];
            edges[j] = edges[j + 1];
            edges[j + 1] = temp;
        }
    }
}

// Initialize disjoint sets
for (int i = 0; i < n; i++)
    parent[i] = i;

printf("\nEdges in the Minimum Spanning Tree (MST):\n");
int mstEdges = 0, totalWeight = 0;

for (int i = 0; i < edgeCount && mstEdges < n - 1; i++) {
    int u = edges[i].src;
    int v = edges[i].dest;

    if (find(u) != find(v)) {
        printf("%d -- %d == %d\n", u, v, edges[i].weight);
        unionSet(u, v);
        totalWeight += edges[i].weight;
        mstEdges++;
    }
}
```

```c
    printf("\nTotal weight of MST = %d\n", totalWeight);
    printf("Minimum edges required to connect all vertices = %d\n",
        mstEdges);

    return 0;
}
```



```
Enter the main string (haystack): hello world
Enter the substring to find (needle): world
Substring found at index 6


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