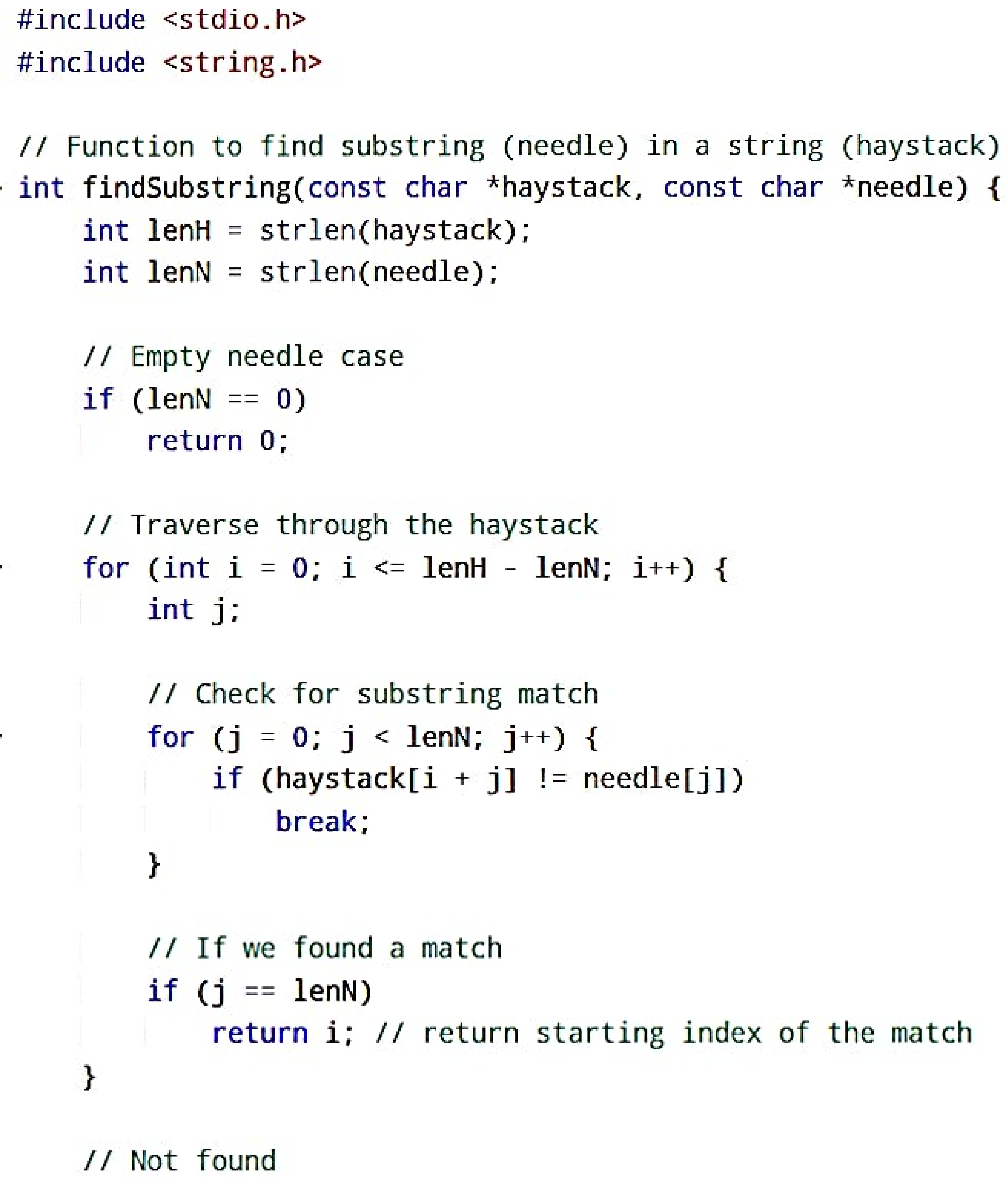
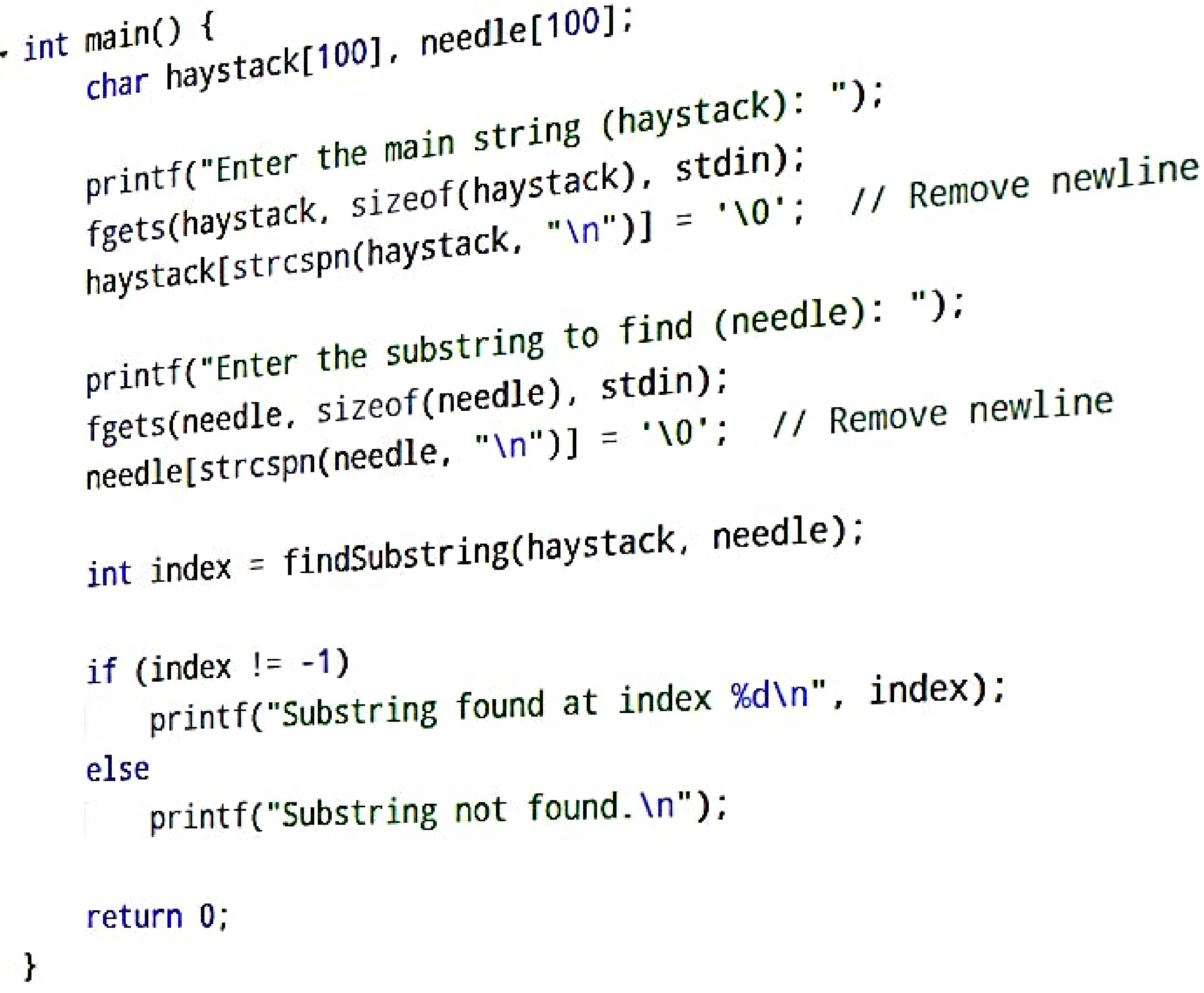
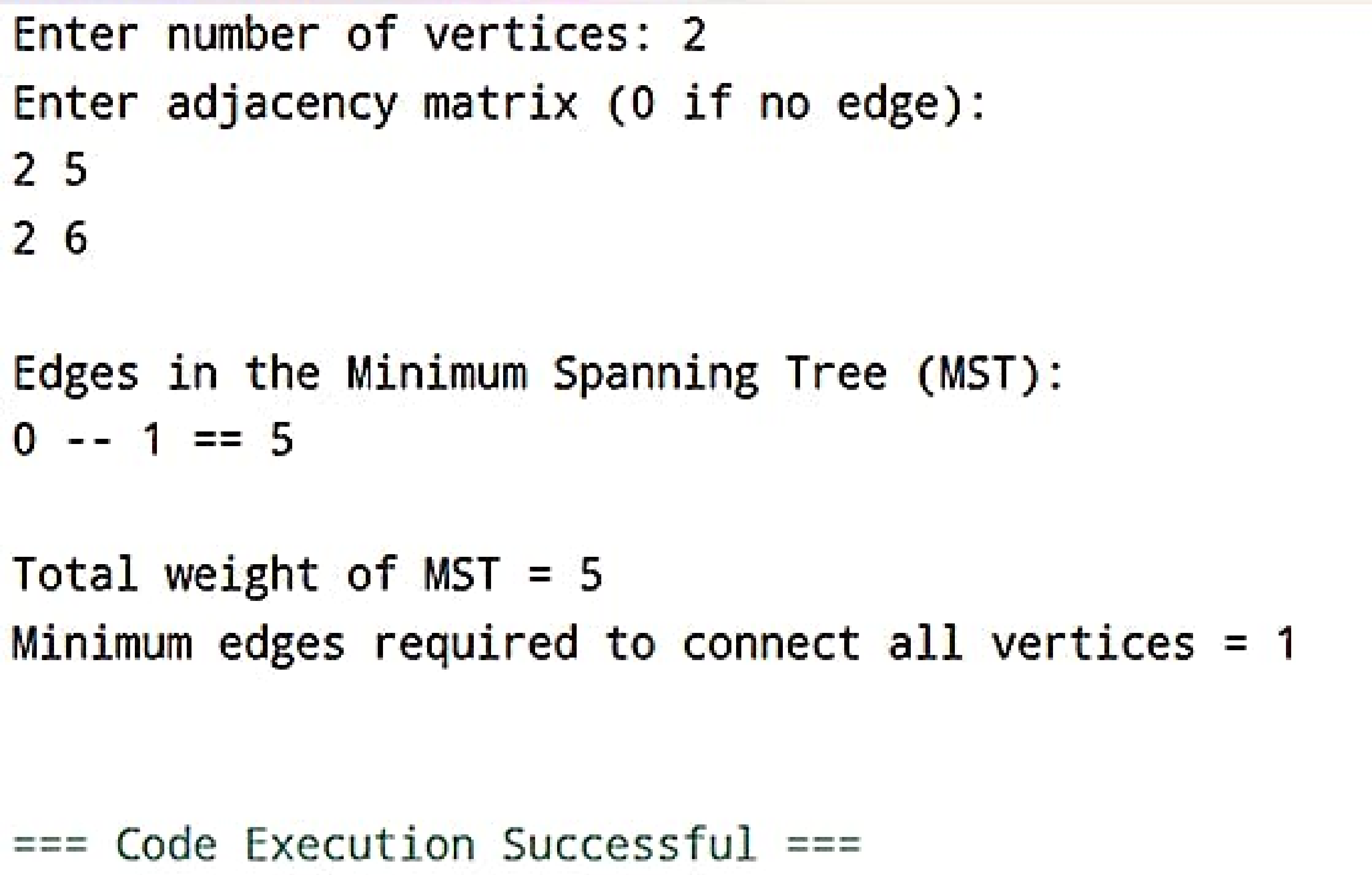


```c
#include <stdio.h>
#include <string.h>

// Function to find substring (needle) in a string (haystack)
int findSubstring(const char *haystack, const char *needle) {
    int lenH = strlen(haystack);
    int lenN = strlen(needle);

    // Empty needle case
    if (lenN == 0)
        return 0;

    // Traverse through the haystack
    for (int i = 0; i <= lenH - lenN; i++) {
        int j;

        // Check for substring match
        for (j = 0; j < lenN; j++) {
            if (haystack[i + j] != needle[j])
                break;
        }

        // If we found a match
        if (j == lenN)
            return i; // return starting index of the match
    }

    // Not found
```



```c
int main() {
    char haystack[100], needle[100];

    printf("Enter the main string (haystack): ");
    fgets(haystack, sizeof(haystack), stdin);
    haystack[strcspn(haystack, "\n")] = '\0';  // Remove newline

    printf("Enter the substring to find (needle): ");
    fgets(needle, sizeof(needle), stdin);
    needle[strcspn(needle, "\n")] = '\0';  // Remove newline

    int index = findSubstring(haystack, needle);

    if (index != -1)
        printf("Substring found at index %d\n", index);
    else
        printf("Substring not found.\n");

    return 0;
}
```



```
Enter number of vertices: 2
Enter adjacency matrix (0 if no edge):
2 5
2 6

Edges in the Minimum Spanning Tree (MST):
0 -- 1 == 5

Total weight of MST = 5
Minimum edges required to connect all vertices = 1


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