ASSIGNMENT NO. 07

TITLE: Breadth First Search (BFS) and Depth First Search (DFS)

NAME: Munajja Mujafar Dalimbkar

PRN: B25CE2011

PROBLEM STATEMENT:

a) Depth First Search (DFS):

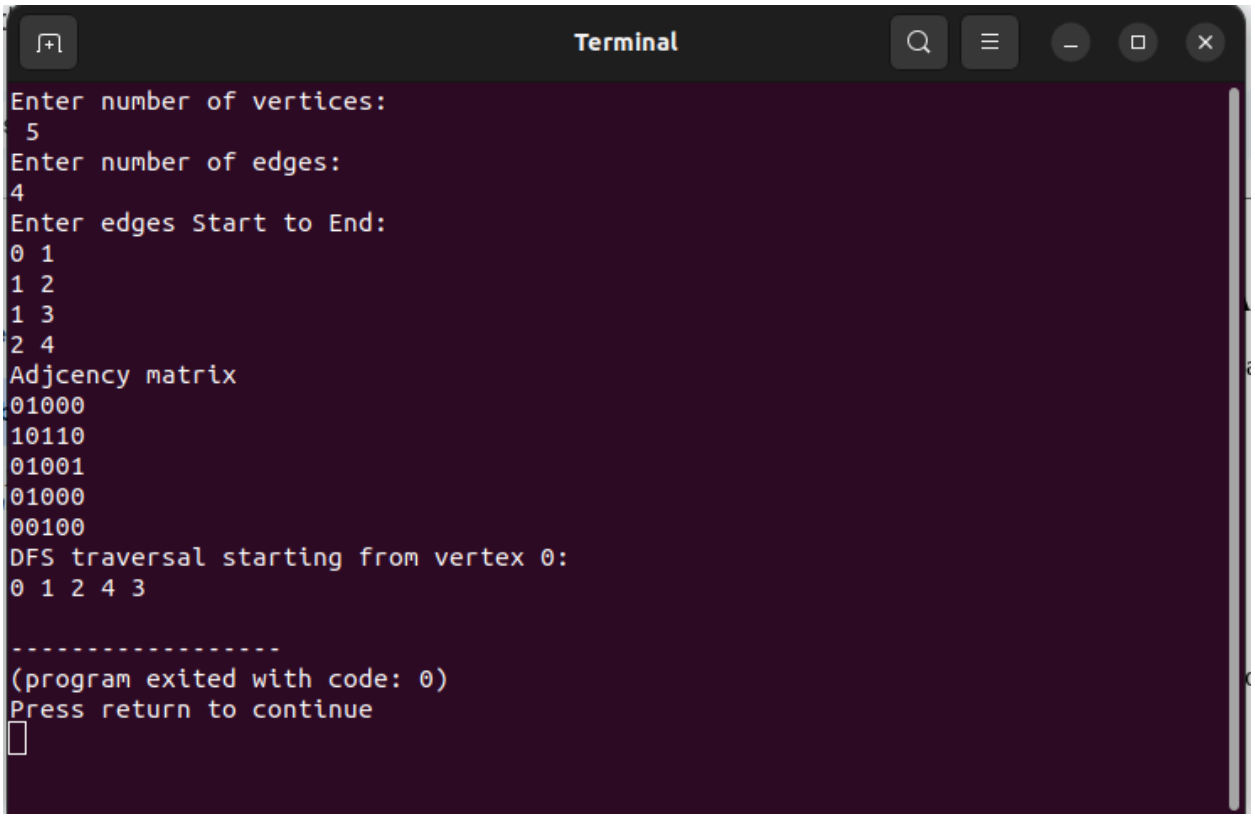
Application: Web crawlers use DFS to explore web pages systematically, following links and indexing content for search engines. Write a simple program to index web pages using Depth First Search (DFS). The program should simulate a web graph where pages are represented as nodes and hyperlinks as edges.

CODE:

```
#include <iostream>

using namespace std;

const int MAX = 100;  // Maximum number of vertices (adjust as needed)

int adj[MAX][MAX];    // Adjacency matrix

bool visited[MAX];    // Visited array

int n;                // Number of vertices

void DFS(int v) {

    visited[v] = true;

    cout << v << " ";

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

        if (adj[v][i] == 1 && !visited[i]) {

            DFS(i);

        }

    }

}
```

```
int main() {
    int edges;
    cout << "Enter number of vertices: ";
    cin >> n;
    cout << "Enter number of edges: ";
    cin >> edges;
    // Initialize adjacency matrix and visited array
    for (int i = 0; i < n; i++) {
        visited[i] = false;
        for (int j = 0; j < n; j++) {
            adj[i][j] = 0;
        }
    }
    // Input edges
    cout << "Enter edges (u v) where u and v are vertex indices starting from 0:\n";
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        adj[u][v] = 1;
        adj[v][u] = 1;  // For undirected graph; remove if directed
    }
    cout << "DFS traversal starting from vertex 0:\n";
    DFS(0);
    return 0;
}
```

OUTPUT:



a) Depth First Search (DFS):

Application: Web crawlers use DFS to explore web pages systematically, following links and indexing content for search engines. Write a simple program to index web pages using Depth First Search (DFS). The program should simulate a web graph where pages are represented as nodes and hyperlinks as edges.

CODE:

```
#include <iostream>

#include <string>

using namespace std;

const int MAX = 100;  // Maximum number of web pages (vertices)

int adj[MAX][MAX];    // Adjacency matrix to represent hyperlinks

bool visited[MAX];    // Track visited pages
```

```
int n;              // Number of web pages (nodes)
// DFS function to simulate indexing
void DFS(int page, string pages[]) {
    visited[page] = true;
    cout << pages[page] << " ";  // Print the current page (indexed)
    for (int i = 0; i < n; i++) {
        if (adj[page][i] == 1 && !visited[i]) {
            DFS(i, pages);
        }
    }
}
int main() {
    int edges;
    cout << "Enter number of web pages: ";
    cin >> n;
    cout << "Enter number of hyperlinks: ";
    cin >> edges;
    string pages[MAX];
    cout << "Enter names of web pages:\n";
    for (int i = 0; i < n; i++) {
        cin >> pages[i];
    }
    // Initialize adjacency matrix and visited array
    for (int i = 0; i < n; i++) {
```

```
        visited[i] = false;
        for (int j = 0; j < n; j++) {
            adj[i][j] = 0;
        }
    }
    // Input hyperlinks (edges)
    cout << "Enter hyperlinks (from to) using page indices (0 to " << n - 1 << "):\n";
    for (int i = 0; i < edges; i++) {
        int u, v;
        cin >> u >> v;
        adj[u][v] = 1;
        adj[v][u] = 1; // For undirected web graph; remove this if directed
    }
    cout << "\nStarting DFS from first web page (" << pages[0] << "):\n";
    DFS(0, pages);
    cout << endl;
    return 0;
}
```

OUTPUT:

```
Enter number of web pages: 5
Enter number of hyperlinks: 5
Enter names of web pages:
Home About Services Contact Team
Enter hyperlinks (from to) using page indices (0 to 4):
0 1
0 2
1 2
2 3
4 3

Starting DFS from first web page (Home):
Home About Services Contact Team
```