ASSIGNMENT-07

/* 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.*/

```
#include <iostream>
using namespace std;

void DFS(int current, int total, bool visited[], int** links) {
    visited[current] = 1;
    cout << "Visiting webpage " << current + 1 << endl;

    for (int i = 0; i < total; i++) {
        if (links[current][i] == 1 && !visited[i]) {
            DFS(i, total, visited, links);
        }
    }
}

int main() {

    int totalPages;
    cout << "Enter the number of webpages: ";
    cin >> totalPages;

    bool* visited = new bool[totalPages];
    int** links = new int*[totalPages];

    for (int i = 0; i < totalPages; i++) {
        links[i] = new int[totalPages];
    }

    for (int i = 0; i < totalPages; i++) {
        for (int j = 0; j < totalPages; j++) {
            links[i][j] = 0;
        }
    }

    for (int i = 0; i < totalPages; i++) {
        for (int j = i + 1; j < totalPages; j++) {

            int connection = 0;
            cout << "Is there a hyperlink between webpage "
                << i + 1 << " and " << j + 1
```

```
            << "? (1 = Yes, 0 = No): ";
        cin >> connection;

        if (connection == 1) {
            links[i][j] = connection;
            links[j][i] = connection;
        }
    }
    cout << endl;
}

cout << "\nAdjacency Matrix (Connection Map):\n";
for (int i = 0; i < totalPages; i++) {
    for (int j = 0; j < totalPages; j++) {
        cout << links[i][j] << " ";
    }
    cout << endl;
}

for (int i = 0; i < totalPages; i++) {
    visited[i] = 0;
}

cout << "\nStarting DFS traversal from webpage 1...\n";
DFS(0, totalPages, visited, links);
cout << "\nDFS traversal complete.\n";

return 0;
}
```

OUTPUT:

```
Enter the number of webpages: 4
Is there a hyperlink between webpage 1 and 2? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 1 and 3? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 1 and 4? (1 = Yes, 0 = No): 0

Is there a hyperlink between webpage 2 and 3? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 2 and 4? (1 = Yes, 0 = No): 0

Is there a hyperlink between webpage 3 and 4? (1 = Yes, 0 = No): 1



Adjacency Matrix (Connection Map):
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0
```

```
Starting DFS traversal from webpage 1...
Visiting webpage 1
Visiting webpage 2
Visiting webpage 3
Visiting webpage 4

DFS traversal complete.


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

/* **Breadth First Search (BFS):**

Application:Indexing web pages for search engines.

Example: A web crawler uses BFS to visit web pages systematically, starting from a seed URL and exploring links level by level. Nodes represent web pages. Edges represent hyperlinks. BFS ensures that pages at the same "depth" (distance from the starting page) are visited before moving to deeper levels. Write a program to simulate the indexing of web pages for a search engine using a Breadth-First Search (BFS) algorithm.

*/

```
#include <iostream>
using namespace std;

void BFS(int start, int total, bool visited[], int** links) {

    int queue[50];
    int front = 0, rear = 0;

    visited[start] = true;
    queue[rear++] = start;

    while (front < rear) {

        int current = queue[front++];
        cout << "Visited Webpage: " << current + 1 << endl;

        for (int i = 0; i < total; i++) {
            if (links[current][i] == 1 && !visited[i]) {
                visited[i] = true;
                queue[rear++] = i;
            }
        }
    }
}

int main() {
```

```
int totalPages;
cout << "Enter Number of Web pages: ";
cin >> totalPages;

bool* visited = new bool[totalPages];
int** links = new int*[totalPages];

for (int i = 0; i < totalPages; i++) {
   links[i] = new int[totalPages];
}

for (int i = 0; i < totalPages; i++) {
   for (int j = 0; j < totalPages; j++) {
      links[i][j] = 0;
   }
}

for (int i = 0; i < totalPages; i++) {
   visited[i] = false;
}

for (int i = 0; i < totalPages; i++) {
   for (int j = i + 1; j < totalPages; j++) {

      int connected = 0;
      cout << "Is there a hyperlink between webpage "
           << i + 1 << " and " << j + 1
           << "? (1 = Yes, 0 = No): ";
      cin >> connected;

      if (connected == 1) {
         links[i][j] = connected;
         links[j][i] = connected;
      }
   }
   cout << endl;
}

cout << "\nAdjacency Matrix (Connection Map):\n";
for (int i = 0; i < totalPages; i++) {
   for (int j = 0; j < totalPages; j++) {
      cout << links[i][j] << " ";
   }
   cout << endl;
}

cout << "\nStarting BFS traversal from webpage 1...\n";
BFS(0, totalPages, visited, links);
```

```
    cout << "\nBFS traversal complete.\n";

    return 0;
}
```

OUTPUT:

```
Enter Number of Web pages: 4
Is there a hyperlink between webpage 1 and 2? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 1 and 3? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 1 and 4? (1 = Yes, 0 = No): 0

Is there a hyperlink between webpage 2 and 3? (1 = Yes, 0 = No): 1
Is there a hyperlink between webpage 2 and 4? (1 = Yes, 0 = No): 0

Is there a hyperlink between webpage 3 and 4? (1 = Yes, 0 = No): 1



Adjacency Matrix (Connection Map):
0 1 1 0
1 0 1 0
1 1 0 1
0 0 1 0

Starting BFS traversal from webpage 1...
Visited Webpage: 1
Visited Webpage: 2
Visited Webpage: 3
Visited Webpage: 4

BFS traversal complete.


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