```cpp
#include <iostream>
#include <queue>
#include <stack>
#define MAX_VERTICES 100
using namespace std;

class MatrixGraph
{
public:
    int V;
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];

    MatrixGraph(int V);

    void addEdgeDirected(int v, int w);
    void addEdgeUndirected(int v, int w);
    void printAdjMatrix();
    void BFS(int start);
    void DFS(int start);
};

MatrixGraph::MatrixGraph(int V)
{
    this->V = V;
    for (int i = 0; i < V; i++)
    {
        for (int j = 0; j < V; j++)
        {
            adjMatrix[i][j] = 0;
        }
    }
}

void MatrixGraph::addEdgeDirected(int v, int w)
{
    adjMatrix[v][w] = 1;
}

void MatrixGraph::addEdgeUndirected(int v, int w)
{
    adjMatrix[v][w] = 1;
    adjMatrix[w][v] = 1;
```

```
}

void MatrixGraph::BFS(int start)
{
    bool visited[MAX_VERTICES] = {false};
    queue<int> q;
    q.push(start);
    int current;
    while (!q.empty())
    {

        current = q.front();
        cout << current << " ";
        for (int i = 0; i < V; i++)
        {
            if (adjMatrix[current][i] && !visited[i])
            {
                visited[i] = true;
                q.push(i);
            }
        }
        q.pop();
    }
    cout << endl;
}
void MatrixGraph::DFS(int start)
{
    bool visited[MAX_VERTICES] = {false};
    stack<int> s;
    s.push(start);
    while (!s.empty())
    {
        int current = s.top();
        s.pop();
        if (!visited[current])
        {
            cout << current << " ";
            visited[current] = true;
            for (int i = 0; i < V; i++)
            {
                if (adjMatrix[current][i] && !visited[i])
                {
```

```
                    s.push(i);
                }
            }
        }
    }
}

void MatrixGraph::printAdjMatrix()
{
    for (int i = 0; i < V; i++)

    {
        for (int j = 0; j < V; j++)
        {
            std::cout << adjMatrix[i][j] << " ";
        }
        std::cout << "\n";
    }
}

int main()
{
    bool arr[5] = {false};
    MatrixGraph g(5);
    g.addEdgeDirected(0, 1);
    g.addEdgeDirected(1, 2);
    g.addEdgeDirected(2, 3);
    g.addEdgeDirected(3, 4);

    g.DFS(0);

    return 0;
}
```