PROGRAM 4

Write program to do the following:

a. Print all the nodes reachable from a given starting node in a digraph using the BFS method.

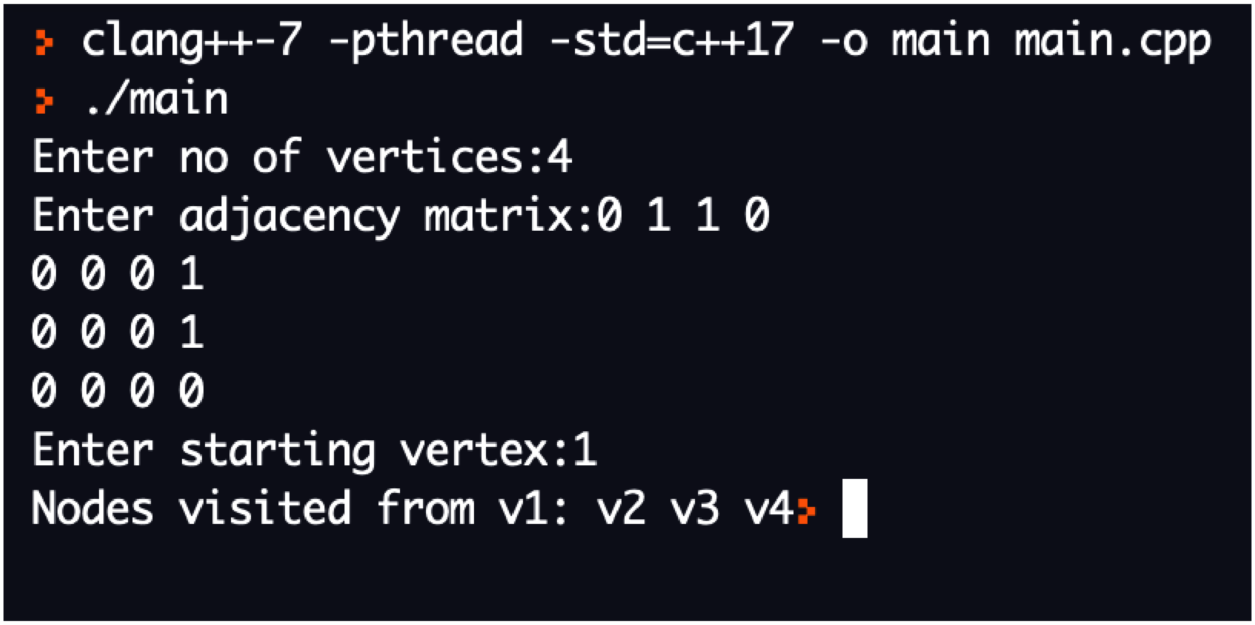
b. Check whether a given graph is connected or not using the DFS method.

```
//BFS Method
#include<iostream>
using namespace std;
int q[10],r=0,f=0;
int a[10][10],visited[10];

void bfs(int v,int n){
  for(int i=0;i<n;i++)
     if(a[v][i] && !visited[i]){
        q[r++]=i;
        visited[i]=1;
     }
  if(f<=r)
     bfs(q[f++],n);
}

int main(){
  int n,k;
  cout<<"Enter no of vertices:";
  cin>>n;
  cout<<"Enter adjacency matrix:";
  for(int i=0;i<n;i++)
     for(int j=0;j<n;j++)
        cin>>a[i][j];
  cout<<"Enter starting vertex:";
  cin>>k;
  visited[k-1]=1;
  for(int i=0;i<n;i++)
     visited[i]=0;
  cout<<"Nodes visited from v"<<k<<":";
  bfs(k-1,n);
  for(int i=0;i<r;i++)
     cout<<" v"<<(q[i]+1);
}
```

//Output



```
› clang++-7 -pthread -std=c++17 -o main main.cpp
› ./main
Enter no of vertices:4
Enter adjacency matrix:0 1 1 0
0 0 0 1
0 0 0 1
0 0 0 0
Enter starting vertex:1
Nodes visited from v1: v2 v3 v4›
```

//DFS Method

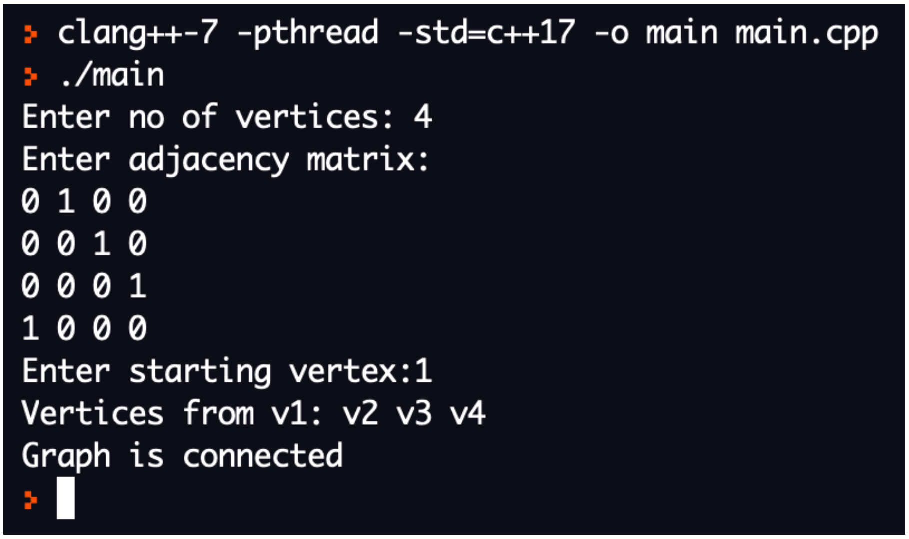
```
#include<iostream>
using namespace std;
int s[10],top=0;
int visited[10],a[10][10];

void dfs(int v,int n){
  s[top++]=v;
  visited[v]=1;
  for(int i=0;i<n;i++){
     if(a[v][i] && !visited[i])
        dfs(i,n);
```

```
    }
}

int main(){
    int n,k;
    cout<<"Enter no of vertices: ";
    cin>>n;
    cout<<"Enter adjacency matrix:";
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)
            cin>>a[i][j];
    cout<<"Enter starting vertex:";
    cin>>k;
    for(int i=0;i<n;i++)
        visited[i]=0;
    visited[k-1]=1;
    dfs(k-1,n);
    cout<<"Vertices from v"<<k<<":";
    for(int i=1;i<top;i++)
        cout<<" v"<<(s[i]+1);
    cout<<endl;
    for(int i=0;i<n;i++)
        if(!visited[i]){
            cout<<"Graph is disconnected"<<endl;
            return 0;
        }
    cout<<"Graph is connected"<<endl;
}
```

//Output



```
› clang++-7 -pthread -std=c++17 -o main main.cpp
› ./main
Enter no of vertices: 4
Enter adjacency matrix:
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
Enter starting vertex:1
Vertices from v1: v2 v3 v4
Graph is connected
› 
```