```cpp
#include<iostream>
#include<vector>
#include<stack>
#include<omp.h>
using namespace std;

const int MAX=10000;
vector<int> graph[MAX];
bool visited[MAX];

void dfs(int node)
{
    stack <int> s;
    s.push(node);
    while(!s.empty())
    {
        int currNode=s.top();
        s.pop();
        if(!visited[currNode]){
            visited[currNode]=true;
            if(visited[currNode])
                cout<<currNode<<" ";
            #pragma omp critical for
            for(int i=0;i<graph[currNode].size();i++)
            {
                int adjNode=graph[currNode][i];
                if(!visited[adjNode])
                    s.push(adjNode);
            }
        }

    }
}
int main()
{
    int n,m,startNode;
    cout<<"\nEmter the no of node: ";
    cin>>n;
    cout<<"\nEnter the no. of edges: ";
    cin>>m;
    cout<<"\nEnter the pair of edges: ";
    for(int i=0;i<m;i++)
    {
        int u,v;
        cin>>u>>v;
        graph[u].push_back(v);
        graph[v].push_back(u);
    }
    #pragma omp parallel for
    for(int i=0;i<n;i++)
    {
        visited[i]=false;
    }
    cout<<"\nEnter the start node: ";
    cin>>startNode;
```

```
56:     dfs(startNode);
57:     return 0;
58: 
59: }
```