WEEK 6

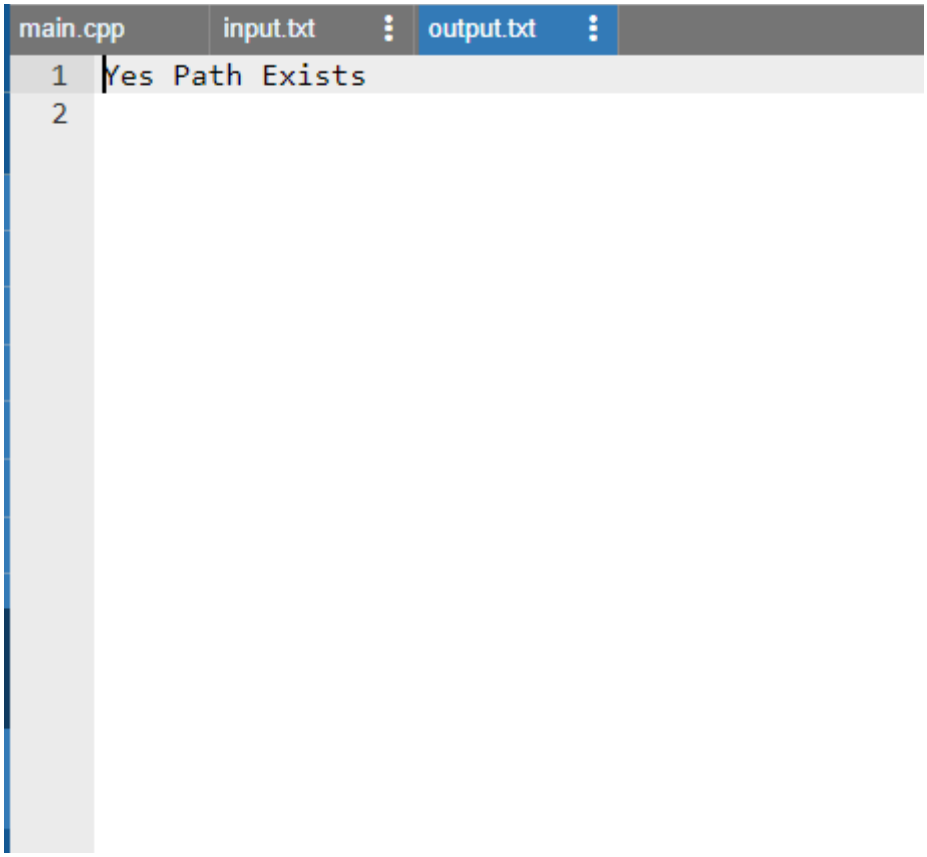
1. Given a (directed/undirected) graph, design an algorithm and implement it using a program to find if a path exists between two given vertices or not. (Hint: use DFS)

```
#include <iostream>
#include <vector>
#include <fstream>
#include<iostream>
using namespace std;
                                                                    //dfs
bool dfs(vector<vector<int>>&adj,vector<int>&visited,int source,int destination){
   if(visited[source]==1) return false;
   visited[source] = 1;
   for(int i:adj[source]){
      if(i==destination) return true;
      if(visited[i]!=1){
         if(dfs(adj,visited,i,destination)) return true;
      }
   }
   return false;
}



int main() {
                                                        // Open input and output files
   ifstream fin("input.txt");
   ofstream fout("output.txt");
                                                    // Check if files are opened successfully
   if (!fin.is_open() || !fout.is_open()) {
      cout << "Error occurred while opening files.\n";
      return 0;
   }
                                                                    //input graph
   int vertices,edges;
```

```
    fin>>vertices>>edges;
    vector<vector<int>>adj(vertices);
    vector<int>visited(vertices);
    for(int i=0;i<edges;i++){
        int u,v;
        fin>>u>>v;
        adj[v].push_back(u);
        adj[u].push_back(v);
    }
    int source,destination;
    fin>>source>>destination;
    if(dfs(adj,visited,source,destination)){
        fout<<"Yes Path Exists\n";
    }
    else{
        fout<<"No Such Path Exists\n";
    }
                                                    // Close input and output files
    fin.close();
    fout.close();
}
```

main.cpp | input.txt | output.txt

```
5 7
0 1
0 2
1 2
1 3
1 4
2 3
3 4
0 4
```


main.cpp
input.txt
output.txt
1 Yes Path Exists
2

2 . Given a graph, design an algorithm and implement it using a program to find if a graph is bipartite or not. (Hint: use BFS)

```
#include <vector>
#include <fstream>
#include<iostream>
#include<queue>
using namespace std;
                                                                    //bfs
bool bfs(vector<vector<int>>&graph,int vertices){
    vector<int>color(vertices,-1);
    queue<int>q;
    color[0]=0;
    q.push(0);
    while(q.empty()==false){
        int node = q.front();
        q.pop();
        for(auto i:graph[node]){
            if(color[i]==-1){
                q.push(i);
                color[i] = !color[node];
            }
            else if(color[i] ==color[node]){
                return false;
            }
        }
    }
    return true;
}

int main() {
                                                    // Open input and output files
    ifstream fin("input.txt");
```

```
    ofstream fout("output.txt");
                                                    // Check if files are opened successfully
    if (!fin.is_open() || !fout.is_open()) {
        cout << "Error occurred while opening files.\n";
        return 0;
    }
                                                                    //input graph
    int vertices,edges;
    fin>>vertices>>edges;
    vector<vector<int>>adj(vertices);

    for(int i=0;i<edges;i++){
        int u,v;
        fin>>u>>v;
        adj[v].push_back(u);
        adj[u].push_back(v);
    }
    if(bfs(adj,vertices)){
        fout<<"Yes Bipartite\n";
    }
    else{
        fout<<"Not Bipartite\n";
    }
                                                        // Close input and output files
    fin.close();
    fout.close();
}
```

main.cpp | input.txt | output.txt

```
8 8
0 1
1 2
2 3
3 4
4 6
6 7
1 7
4 5
```

main.cpp | input.txt | output.txt

```
1  Yes Bipartite
2
```

3 . Given a directed graph, design an algorithm and implement it using a program to find whether cycle exists in the graph or not.

```
#include <vector>
#include <fstream>
#include<iostream>
#include<queue>
using namespace std;
                                                                    //bfs
bool bfs(vector<vector<int>>&graph,int vertices){
    vector<int>visited(vertices,-1);
    queue<pair<int,int>>q;
    visited[0]=1;
    q.push({0,-1});
    while(q.empty()==false){
        int node = q.front().first;
        int parent = q.front().second;
        q.pop();
        for(auto i:graph[node]){
            if(visited[i]==-1){
                q.push({i,node});
                visited[i] = !visited[node];
            }
            else if(i!=parent) return true;
        }
    }
    return false;
}

int main() {
                                                        // Open input and output files
    ifstream fin("input.txt");
    ofstream fout("output.txt");
    // Check if files are opened successfully
    if (!fin.is_open() || !fout.is_open()) {
```

```
        cout << "Error occurred while opening files.\n";
        return 0;
    }
                                                                    //input graph
    int vertices,edges;
    fin>>vertices>>edges;
    vector<vector<int>>adj(vertices);

    for(int i=0;i<edges;i++){
        int u,v;
        fin>>u>>v;
        adj[v].push_back(u);
        adj[u].push_back(v);
    }
    if(bfs(adj,vertices)){
        fout<<"Yes Cycle Exists\n";
    }
    else{
        fout<<"No Cycle Exists\n";
    }
                                                    // Close input and output files
    fin.close();
    fout.close();
}
```

main.cpp | input.txt | output.txt

```
8 8
0 1
1 2
2 3
3 4
4 6
6 7
1 7
4 5
```

main.cpp | input.txt | output.txt

```
1  Yes Cycle Exists
2
```