```cpp
#include<iostream>
#include<vector>
#include<stack>
#include<unordered_map>
using namespace std;
class Graph{

    int vertex;
    vector<unordered_map<int,int>> adj;

    public:
        Graph(int v){
            vertex = v;
            adj = vector<unordered_map<int,int>>(v+1);
        }
        void addEdge(int u, int v){
            adj[u][v] = 1;
            adj[v][u] = 1;
        }
        void removeEdge(int v,int u){
            adj[v].erase(u);
            adj[u].erase(v);
        }

        // function checks if the graph contains a euler
path/circuit or not
        void printEulerPathCircuit(){

            int odd = 0; // number of vertices with odd degree
            int oddVertex = 0; // it stores vertex with odd
degree if it exists

            for(int i=1;i<=vertex;++i){
                if(adj[i].size()%2==1){
                    ++odd;
                    oddVertex = i;
                }
```

```
        }

        if(odd==0){
            cout<<"Euler Circuit: ";
            printEuler(1);
        }
        else if(odd==2){
            cout<<"Euler Path: ";
            printEuler(oddVertex);
        }
        else{
            cout<<"Euler Path/Circuit Doesn't Exist"<<endl;
        }

    }
    void printEuler(int v){

        stack<int> cpath;    // current path
        stack<int> epath;    // euler path

        cpath.push(v);        // euler path starts from v

        while(!cpath.empty()){
            int u = cpath.top();

            if(adj[u].size()==0){
                epath.push(u);
                cpath.pop();
            }
            else{
                cpath.push(adj[u].begin()->first);
                removeEdge(u,adj[u].begin()->first);
            }
        }

        while(!epath.empty()){
            cout<<" "<<epath.top()<<" ";
```

```
                epath.pop();
            }

        }

};
int main()
{
    int v=0;
    cout << "Enter number of vertices: ";
    cin >> v;
    Graph G(v);
    // G.addEdge(1, 6);
    // G.addEdge(6, 3);
    // G.addEdge(3, 2);
    // G.addEdge(2, 1);
    // G.addEdge(2, 5);
    // G.addEdge(5, 4);
    // G.addEdge(4, 2);
    int i = 0;
    int j =0;
    cout << "Input edges, input -1 to either to finish input
(only input in range" << endl;
    while(1){
      cout << "Input from vertex: ";
      cin >> i;
      cout << "Input to vertext: ";
      cin >> j;
      if(i == -1 || j==-1){
        break;
      }
      cout << "Added edge from " << i << " to " << j << endl;
      G.addEdge(i,j);
    }
    G.printEulerPathCircuit();
}
```

Output:

```
Enter number of vertices: 6
Input edges, input -1 to either to finish input
Input from vertex: 1
Input to vertext: 6
Added edge from 1 to 6
Input from vertex: 6
Input to vertext: 3
Added edge from 6 to 3
Input from vertex: 3
Input to vertext: 2
Added edge from 3 to 2
Input from vertex: 2
Input to vertext: 1
Added edge from 2 to 1
Input from vertex: 2
Input to vertext: 5
Input from vertex: 5
Input to vertext: 4
Added edge from 5 to 4
Input from vertex: 4
Input to vertext: 2
Added edge from 4 to 2
Input from vertex: -1
Input to vertext: 0
Euler Circuit:  1  2  4  5  2  3  6  1
```