BÁCH KHOA E-LEARNING

Câu hỏi **8**

Không hoàn thành

Chấm điểm của 1,00

Implement Depth-first search

```
Adjacency *DFS(int v);
```

where Adjacency is a structure to store list of number.

```
#include <iostream>
#include <list>
using namespace std;

class Adjacency
{
private:
        list<int> adjList;
        int size;
public:
        Adjacency() {}
        Adjacency(int V) {}
        void push(int data)
        {
                adjList.push_back(data);
                size++;
        }
        void print()
        {
                for (auto const &i : adjList)
                        cout << " -> " << i;
        }
        void printArray()
        {
                for (auto const &i : adjList)
                        cout << i << " ";
        }
        int getSize() { return adjList.size(); }
        int getElement(int idx)
        {
                auto it = adjList.begin();
                advance(it, idx);
                return *it;
        }
};
```

And Graph is a structure to store a graph (see in your answer box)

For example:

Test	Result

`int V = 8, visited = 0;`
 	
`Graph g(V);`
`Adjacency *arr;`
`int edge[][2] = {{0,1}, {0,2}, {0,3}, {0,4}, {1,2}, {2,5}, {2,6}, {4,6}, {6,7}};`
`for(int i = 0; i < 9; i++)`
`{`
`  g.addEdge(edge[i][0], edge[i][1]);`
`}`
 	
`// g.printGraph();`
`// cout << endl;`
`arr = g.DFS(visited);`
`arr->printArray();`
`delete arr;`
	`0 1 2 5 6 4 7 3`

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph
{
private:
    int V;
    Adjacency *adj;

public:
    Graph(int V)
    {
        this->V = V;
        adj = new Adjacency[V];
    }

    void addEdge(int v, int w)
    {
        adj[v].push(w);
        adj[w].push(v);
    }

    void printGraph()
    {
        for (int v = 0; v < V; ++v)
        {
            cout << "\nAdjacency list of vertex " << v << "\nhead ";
            adj[v].print();
        }
    }

    Adjacency *DFS(int v)
    {
        // v is a vertex we start DFS
    }
};
```

Kiểm tra

◀ Practice Exercises

Chuyển tới...

Discussion ►