Đã bắt đầu vào lúc	Thứ bảy, 9 Tháng mười hai 2023, 12:51 PM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ bảy, 9 Tháng mười hai 2023, 3:54 PM
Thời gian thực hiện	3 giờ 2 phút
Điểm	**10,00** của 10,00 (**100**%)

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Implement Breadth-first search

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

int V = 6;
int visited = 2;
 	
Graph g(V);
Adjacency* arr = new Adjacency(V);
int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}};
 	
for(int i = 0; i < 8; i++)
{
    g.addEdge(edge[i][0], edge[i][1]);
}
 	
arr = g.BFS(visited);
arr->printArray();
delete arr;
	2 0 4 1 3 5

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 *BFS(int v) {
        // v is a vertex we start BFS
    bool* visited = new bool[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    Adjacency* result = new Adjacency(V);

    int* queue = new int[V];
    int front = 0, rear = 0;

    visited[v] = true;
    queue[rear++] = v;

    while (front != rear)
    {
        int currVertex = queue[front++];
        result->push(currVertex);

        for (int i = 0; i < adj[currVertex].getSize(); i++)
        {
            int nextVertex = adj[currVertex].getElement(i);
            if (!visited[nextVertex])
            {
                visited[nextVertex] = true;
                queue[rear++] = nextVertex;
            }
        }
    }

    delete[] visited;
    delete[] queue;
    return result;
    }
};
```

	Test	Expected	Got	

✓	int V = 6;
	int visited = 0;
	
	Graph g(V);
	Adjacency* arr = new Adjacency(V);
	int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}};
	
	for(int i = 0; i < 8; i++)
	{
	    g.addEdge(edge[i][0], edge[i][1]);
	}
	
	arr = g.BFS(visited);
	arr->printArray();
	delete arr;
		0 1 2 3 4 5	0 1 2 3 4 5	✓

	Test	Expected	Got	

✓	int V = 6;
	int visited = 2;
	
	Graph g(V);
	Adjacency* arr = new Adjacency(V);
	int edge[][2] = {{0,1},{0,2},{1,3},{1,4},{2,4},{3,4},{3,5},{4,5}};
	
	for(int i = 0; i < 8; i++)
	{
	    g.addEdge(edge[i][0], edge[i][1]);
	}
	
	arr = g.BFS(visited);
	arr->printArray();
	delete arr;
		2 0 4 1 3 5	2 0 4 1 3 5	✓

✓	int V = 8, visited = 5;
	
	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++)
	{
	\tg.addEdge(edge[i][0], edge[i][1]);
	}
	
	// g.printGraph();
	// cout << endl;
	arr = g.BFS(visited);
	arr->printArray();
	delete arr;
		5 2 0 1 6 3 4 7	5 2 0 1 6 3 4 7	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Chính xác

Điểm 1,00 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)
{
    bool* visited = new bool[V];
    for (int i = 0; i < V; i++)
        visited[i] = false;

    Adjacency* result = new Adjacency(V);
    DFSUtil(v, visited, *result);

    delete[] visited;
    return result;
}

void DFSUtil(int v, bool* visited, Adjacency& result)
{
    visited[v] = true;
    result.push(v);

    for (int i = 0; i < adj[v].getSize(); i++)
    {
        int nextVertex = adj[v].getElement(i);
        if (!visited[nextVertex])
            DFSUtil(nextVertex, visited, result);
    }
}

};
```

	Test	Expected	Got	

✓	`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++)`
	`{`
	`\tg.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`	`0 1 2 5 6 4 7 3`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 3

Chính xác

Điểm 1,00 của 1,00

Given a graph represented by an adjacency-list `edges`.

Request: Implement function:

```
int connectedComponents(vector<vector<int>>& edges);
```

Where `edges` is the adjacency-list representing the graph (this list has between 0 and 1000 lists). This function returns the number of connected components of the graph.

Example:

Given a adjacency-list: `[[1], [0, 2], [1], [4], [3], []]`

There are `3` connected components: `[0, 1, 2], [3, 4], [5]`

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

For example:

Test	Result

vector<vector<int>> graph {
        {1},
        {0, 2},
        {1, 3},
        {2},
        {}
};
cout << connectedComponents(graph);
	2

Answer: (penalty regime: 0 %)

Reset answer

```
int DFS(vector<vector<int>>& edges, int vertex, vector<bool>& visited) {
    visited[vertex] = true;
    int count = 1;

    for (int neighbor : edges[vertex]) {
        if (!visited[neighbor]) {
            count += DFS(edges, neighbor, visited);
        }
    }

    return count;
}

int connectedComponents(vector<vector<int>>& edges) {
    int n = edges.size();
    vector<bool> visited(n, false);
    int components = 0;

    for (int i = 0; i < n; i++) {
        if (!visited[i]) {
            components++;
            DFS(edges, i, visited);
        }
    }

    return components;
}
```

	Test	Expected	Got	

✓	`vector<vector<int>> graph {`
	`\t{1},`
	`\t{0, 2},`
	`\t{1, 3},`
	`\t{2},`
	`\t{}`
	`};`
	`cout << connectedComponents(graph);`
		2	2	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 4

Chính xác

Điểm 1,00 của 1,00

Implement function to detect a cyclic in Graph

```
bool isCyclic();
```

Graph structure is defined in the initial code.

For example:

Test	Result

`DirectedGraph g(8);`
`int edege[][2] = {{0,6}, {1,2}, {1,4}, {1,6}, {3,0}, {3,4}, {5,1}, {7,0}, {7,1}};`
 	
`for(int i = 0; i < 9; i++)`
`        g.addEdge(edege[i][0], edege[i][1]);`
 	
`if(g.isCyclic())`
`        cout << "Graph contains cycle";`
`else`
`        cout << "Graph doesn't contain cycle";`
	`Graph doesn't contain cycle`

Answer: (penalty regime: 0 %)

Reset answer

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

class DirectedGraph
{
    int V;
    vector<list<int>> adj;
public:
    DirectedGraph(int V)
    {
        this->V = V;
        adj = vector<list<int>>(V, list<int>());
    }
    void addEdge(int v, int w)
    {
        adj[v].push_back(w);
    }
bool isCyclic()
{
    vector<bool> visited(V, false);
    vector<bool> recursionStack(V, false);

    for (int v = 0; v < V; v++)
    {
        if (!visited[v])
        {
            if (isCyclicUtil(v, visited, recursionStack))
                return true;
        }
    }

    return false;

}
bool isCyclicUtil(int v, vector<bool>& visited, vector<bool>& recursionStack)
{
    visited[v] = true;
    recursionStack[v] = true;

    for (auto neighbor : adj[v])
    {
        if (!visited[neighbor])
        {
            if (isCyclicUtil(neighbor, visited, recursionStack))
                return true;
        }
        else if (recursionStack[neighbor])
        {
            return true;
        }
    }

    recursionStack[v] = false;
    return false;
}
};
```

	Test	Expected	Got	

✓	`DirectedGraph g(8);`
	`int edege[][2] = {{0,6}, {1,2}, {1,4}, {1,6}, {3,0}, {3,4}, {5,1}, {7,0}, {7,1}};`
	
	`for(int i = 0; i < 9; i++)`
	`\tg.addEdge(edege[i][0], edege[i][1]);`
	
	`if(g.isCyclic())`
	`\tcout << "Graph contains cycle";`
	`else`
	`\tcout << "Graph doesn't contain cycle";`
		Graph doesn't contain cycle	Graph doesn't contain cycle	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 5

Chính xác

Điểm 1,00 của 1,00

Given a graph and a source vertex in the graph, find shortest paths from source to destination vertice in the given graph using Dijsktra's algorithm.

Following libraries are included: iostream, vector, algorithm, climits, queue

For example:

Test	Result

int n = 6;
int init[6][6] = {
        {0, 10, 20, 0, 0, 0},
        {10, 0, 0, 50, 10, 0},
        {20, 0, 0, 20, 33, 0},
        {0, 50, 20, 0, 20, 2},
        {0, 10, 33, 20, 0, 1},
        {0, 0, 0, 2, 1, 0} };
 	
int** graph = new int*[n];
for (int i = 0; i < n; ++i) {
        graph[i] = init[i];
}
 	
cout << Dijkstra(graph, 0, 1);
	10

Answer: (penalty regime: 0 %)

Reset answer

```
// Some helping functions

int MinDist(int* dist, bool* spt, int node) {
    int min = INT_MAX;
    int min_index;

    for (int i = 0; i < node; i++) {
        if (!spt[i] && dist[i] <= min) {
            min = dist[i];
            min_index = i;
        }
    }
    return min_index;
}

int Dijkstra(int** graph, int src, int dst) {
    int NumVertices =   6 ;

    int* dist = new int[NumVertices];
    bool* spt = new bool[NumVertices];

    for (int i = 0; i < NumVertices; i++) {
        dist[i] = INT_MAX;
        spt[i] = false;
    }
    dist[src] = 0;

    for (int i = 0; i < NumVertices - 1; i++) {
        int u = MinDist(dist, spt, NumVertices);
        spt[u] = true;

        for (int v = 0; v < NumVertices; v++) {
            if (!spt[v] && graph[u][v] && dist[u] != INT_MAX &&
                dist[u] + graph[u][v] < dist[v]) {
```

```
                dist[v] = dist[u] + graph[u][v];
            }
        }
    }

    int result = dist[dst];
    return result;
}

```

	Test	Expected	Got	

✓	`int n = 6;`
	`int init[6][6] = {`
	`\t{0, 10, 20, 0, 0, 0},`
	`\t{10, 0, 0, 50, 10, 0},`
	`\t{20, 0, 0, 20, 33, 0},`
	`\t{0, 50, 20, 0, 20, 2},`
	`\t{0, 10, 33, 20, 0, 1},`
	`\t{0, 0, 0, 2, 1, 0} };`
	
	`int** graph = new int*[n];`
	`for (int i = 0; i < n; ++i) {`
	`\tgraph[i] = init[i];`
	`}`
	
	`cout << Dijkstra(graph, 0, 1);`
		10	10	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 6

Chính xác

Điểm 1,00 của 1,00

Implement **topologicalSort** function on a graph. (Ref here)

```
void topologicalSort();
```

where Adjacency is a structure to store list of number. Note that, the vertex index starts from 0. **To match the given answer, please always traverse from 0 when performing the sorting.**

```
#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). You could write one or more helping functions.

For example:

Test	Result

`Graph g(6);`
`g.addEdge(5, 2);`
`g.addEdge(5, 0);`
`g.addEdge(4, 0);`
`g.addEdge(4, 1);`
`g.addEdge(2, 3);`
`g.addEdge(3, 1);`
 	
`g.topologicalSort();`
	`5 4 2 3 1 0`

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph {

    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);
    }

    //Heling functions

    void topologicalSortUtil(int vertice, bool visited[], stack<int>& Stk){
        visited[vertice] = true;
        for(int i = 0; i < adj[vertice].getSize(); i++){
            int temp = adj[vertice].getElement(i);
            if(!visited[temp]){
                topologicalSortUtil(temp, visited, Stk);
            }
        }
        Stk.push(vertice);
    }
    void topologicalSort(){
        //TODO
        stack<int> stk;
        bool* visited = new bool[V];
        for(int i = 0; i < V; i++){
            visited[i] = false;
        }
        for(int i = 0; i < V; i++){
            if(!visited[i]){
                topologicalSortUtil(i, visited, stk);
            }
        }
        while(!stk.empty()){
            cout << stk.top() << " ";
            stk.pop();
        }
    }
};
```

	Test	Expected	Got	

✓	`Graph g(6);`
	`g.addEdge(5, 2);`
	`g.addEdge(5, 0);`
	`g.addEdge(4, 0);`
	`g.addEdge(4, 1);`
	`g.addEdge(2, 3);`
	`g.addEdge(3, 1);`
	
	`g.topologicalSort();`
		`5 4 2 3 1 0`	`5 4 2 3 1 0`	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 7

Chính xác

Điểm 1,00 của 1,00

Implement function

```
int foldShift(long long key, int addressSize);
int rotation(long long key, int addressSize);
```

to hashing key using Fold shift or Rotation algorithm.

Review Fold shift:

The **folding method** for constructing hash functions begins by dividing the item into equal-size pieces (the last piece may not be of equal size). These pieces are then added together to give the resulting hash value.

For example:

Test	Result
`cout << rotation(600101, 2);`	26

Answer: (penalty regime: 0 %)

Reset answer

```
int foldShift(long long key, int addressSize)
{
    long long res = 0;
    string s = to_string(key);
    for (unsigned int i = 0; i < s.size(); i += addressSize) {
        res += stoll(s.substr(i,addressSize));
    }
    string a = to_string(res);
    return stoi(a.substr(a.size() - addressSize));
}

int rotation(long long key, int addressSize)
{
    string s = to_string(key);
    s = s[s.size() - 1] + s.substr(0, s.size() - 1);
    int newkey = foldShift(stoll(s), addressSize);
    return newkey;
}
```

	Test	Expected	Got	
✓	`cout << rotation(600101, 2);`	26	26	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 8

Chính xác

Điểm 1,00 của 1,00

The relationship between a group of people is represented by an adjacency-list `friends`. If `friends[u]` contains `v`, `u` and `v` are friends.Friendship is a two-way relationship. Two people are in a friend group as long as there is some path of mutual friends connecting them.

Request: Implement function:

```
int numberOfFriendGroups(vector<vector<int>>& friends);
```

Where `friends` is the adjacency-list representing the friendship (this list has between 0 and 1000 lists). This function returns the number of friend groups.

Example:

Given a adjacency-list: `[[1], [0, 2], [1], [4], [3], []]`

There are `3` friend groups: `[0, 1, 2], [3, 4], [5]`

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` have been included and `namespace std` is used. You can write helper functions and class. Importing other libraries is allowed, but not encouraged.

For example:

Test	Result

vector<vector<int>> graph {
        {1},
        {0, 2},
        {1},
        {4},
        {3},
        {}
};
cout << numberOfFriendGroups(graph);
	3

Answer: (penalty regime: 0 %)

Reset answer

```
void DFS(int node, const std::vector<std::vector<int>>& friends, std::unordered_set<int>& visited) {
    visited.insert(node);

    for (int friendNode : friends[node]) {
        if (visited.find(friendNode) == visited.end()) {
            DFS(friendNode, friends, visited);
        }
    }
}

int numberOfFriendGroups(vector<vector<int>>& friends) {
    // STUDENT ANSWER
    int n = friends.size();
    unordered_set<int> visited;
    int count = 0;

    for (int i = 0; i < n; ++i) {
        if (visited.find(i) == visited.end()) {
            DFS(i, friends, visited);
            ++count;
        }
    }

    return count;
}
```

	Test	Expected	Got	

✓	vector<vector<int>> graph {
	\t{1},
	\t{0, 2},
	\t{1},
	\t{4},
	\t{3},
	\t{}
	};
	cout << numberOfFriendGroups(graph);
		3	3	✓

✓	vector<vector<int>> graph {
	
	};
	cout << numberOfFriendGroups(graph);
		0	0	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 9

Chính xác

Điểm 1,00 của 1,00

Implement three following hashing function:

```
long int midSquare(long int seed);
long int moduloDivision(long int seed, long int mod);
long int digitExtraction(long int seed, int* extractDigits, int size);
```

Note that:

In midSquare function: we eliminate 2 last digits and get the 4 next digits.

In digitExtraction: extractDigits is a sorted array from smallest to largest index of digit in seed (index starts from 0). The array has size **size.**

For example:

Test	Result

int a[]={1,2,5};
cout << digitExtraction(122443,a,3);
	223

cout <<midSquare(9452);	3403

Answer: (penalty regime: 0 %)

Reset answer

```
long int midSquare(long int seed) {
    long int square = seed * seed;
    long int mid = (square / 100) % 10000;
    return mid;
}

long int moduloDivision(long int seed, long int mod) {
    return seed % mod;
}

long int digitExtraction(long int seed, int* extractDigits, int size) {
    long int result = 0;
    int numDigits = floor(log10(seed)) + 1;
    for (int i = 0; i < size; i++) {
        int digit = numDigits - extractDigits[i] - 1;
        int extractedDigit = seed / static_cast<long int>(pow(10, digit)) % 10;
        result = result * 10 + extractedDigit;
    }
    return result;
}

/*int reverseInteger(int num) {
    int reversed = 0;

    while (num != 0) {
        int digit = num % 10;
        reversed = reversed * 10 + digit;
        num /= 10;
    }

    return reversed;
}

long int digitExtraction(long int seed, int* extractDigits, int size)
{
    vector<int> tmp(extractDigits, extractDigits + size);
    int tmptmp = seed;
```

	Test	Expected	Got	

✓	`int a[]={1,2,5};`
	`cout << digitExtraction(122443,a,3);`
		223	223	✓

✓	`cout <<midSquare(9452);`	3403	3403	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 10

Chính xác

Điểm 1,00 của 1,00

There are `n` people, each person has a number between `1` and `100000` (`1 ≤ n ≤ 100000`). Given a number `target`. Two people can be matched as a **perfect pair** if the sum of numbers they have is equal to `target`. A person can be matched no more than 1 time.

Request: Implement function:

```
int pairMatching(vector<int>& nums, int target);
```

Where `nums` is the list of numbers of `n` people, `target` is the given number. This function returns the number of **perfect pairs** can be found from the list.

Example:

The list of numbers is `{1, 3, 5, 3, 7}` and `target = 6`. Therefore, the number of **perfect pairs** can be found from the list is `2` (pair `(1, 5)` and pair `(3, 3)`).

Note:

In this exercise, the libraries `iostream, string, cstring, climits, utility, vector, list, stack, queue, map, unordered_map, set, unordered_set, functional, algorithm` has been included and `namespace std` are used. You can write helper functions and classes. Importing other libraries is allowed, but not encouraged, and may result in unexpected errors.

For example:

Test	Result

`vector<int>items{1, 3, 5, 3, 7};`
`int target = 6;`
`cout << pairMatching(items, target);`
	2

`int target = 6;`
`vector<int>items{4,4,2,1,2};`
`cout << pairMatching(items, target);`
	2

Answer: (penalty regime: 0 %)

Reset answer

```
int pairMatching(vector<int>& nums, int target) {
    unordered_map<int, int> freq;
    int pairs = 0;

    for (int num : nums) {
        int complement = target - num;


        if (freq.find(complement) != freq.end() && freq[complement] > 0) {
            pairs++;
            freq[complement]--;
        } else {
            freq[num]++;
        }
    }

    return pairs;
}
```

	Test	Expected	Got	

✓	vector<int>items{1, 3, 5, 3, 7};
	int target = 6;
	cout << pairMatching(items, target);
		2	2	✓

Passed all tests! ✓

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

BÁCH KHOA E-LEARNING

WEBSITE

HCMUT

MyBK

BKSI

LIÊN HỆ

268 Lý Thường Kiệt, P.14, Q.10, TP.HCM

☎ (028) 38 651 670 - (028) 38 647 256 (Ext: 5258, 5234)

elearning@hcmut.edu.vn