BÁCH KHOA E-LEARNING

Thời gian còn lại 0:59:49

Câu hỏi **1**

Không hoàn thành

Chấm điểm của 3,00

void reduceDuplicate(Node* root);

To reduce the continuous duplicate.
For example, given the linked list 122234452, the output will be the linked list 123452. (the final 2 is not removed because they are not continuous to the previous group of 2s).

Given the Node defined as the following

```
class Node
{
    int data;
    Node* next;
public:
    Node(): data(0), next(nullptr){}

    Node(int data, Node* next)
    {
        this->data = data;
        this->next = next;
    }

    int getData()
    {
        return this->data;
    }

    void setData(int data)
    {
        this->data = data;
    }

    Node* getNext()
    {
        return this->next;
    }

    void setNext(Node* next)
    {
        this->next = next;
    }
};
```

For example:

Test	Result

`Node* node1 = new Node(1, nullptr);`
`Node* node2 = new Node(1, node1);`
`Node* node3 = new Node(0, node2);`
`printList(node3);`
`reduceDuplicate(node3);`
`printList(node3);`
	`HEAD -> 0 -> 1 -> 1 -> NULL`
	`HEAD -> 0 -> 1 -> NULL`

Answer: (penalty regime: 0 %)

Reset answer

```
1 void reduceDuplicate(Node* root)
```

```
{
}
```

Kiểm tra

Câu hỏi **2**

Không hoàn thành

Chấm điểm của 3,00

Given the template for class PrinterQueue and 2 methods:

1. addNewRequest(int priority, string fileName)

Add a file to queue with **priority** and **fileName.** The test cases have maximum of 100 file.

2. print()
Print the highest priority file's fileName. If there is no file in the queue, print "No file to print";

- The files that have the same priority will be printed in FIFO (First In First Out) order

You can implement any additional method.

For example:

Test	Result

PrinterQueue* myPrinterQueue = new PrinterQueue();
myPrinterQueue->addNewRequest(1, "hello.pdf");
myPrinterQueue->addNewRequest(2, "goodbye.pdf");
myPrinterQueue->addNewRequest(2, "goodnight.pdf");
myPrinterQueue->print();
myPrinterQueue->print();
myPrinterQueue->print();
	goodbye.pdf
	goodnight.pdf
	hello.pdf

PrinterQueue* myPrinterQueue = new PrinterQueue();
myPrinterQueue->addNewRequest(1, "hello.pdf");
myPrinterQueue->print();
myPrinterQueue->print();
myPrinterQueue->print();
	hello.pdf
	No file to print
	No file to print

Answer: (penalty regime: 0 %)

Reset answer

```
class PrinterQueue
{
    // your attributes
public:
    // your methods

    void addNewRequest(int priority, string fileName)
    {
        // your code here
    }

    void print()
    {
        // your code here
        // After some logic code, you have to print fileName with endline
    }
};
```

Kiểm tra

Câu hỏi **3**

Không hoàn thành

Chấm điểm của 4,00

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.

The function will return the shortest distance and the **Path** variables will be the shortest path from src to dst.

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];
}
int Path[7];
cout << Dijkstra(graph, 0, 0,Path)<<"\n";
for(int i=0;Path[i]!=-1;i++){
cout<<Path[i]<<" ";
}
	0
	0

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];
}
 	
int Path[7];
cout << Dijkstra(graph, 0, 1,Path)<<"\n";
for(int i=0;Path[i]!=-1;i++){
cout<<Path[i]<<" ";
}
	10
	0

Answer: (penalty regime: 0 %)

Reset answer

```
// Some helping functions

int* Dijkstra(int** graph, int src, int dst,int Path[]) {
    // TODO: return the length of shortest path from src to dst.

}

```

Kiểm tra

◄ Test at 9 or 10AM

Chuyển tới...

Week1(re-Practice) ►