Đã bắt đầu vào lúc	Thứ tư, 23 Tháng mười một 2022, 8:14 AM
Tình trạng	Đã hoàn thành
Hoàn thành vào lúc	Thứ tư, 23 Tháng mười một 2022, 8:57 AM
Thời gian thực hiện	42 phút 59 giây
Điểm	1,10/12,00
Điểm	**0,92** của 10,00 (**9,17**%)
	Theo CBM, điểm trên được hiển thị liên quan tối đa cho tất cả chính xác tại C=1.
	Kết quả cho toàn bộ câu hỏi bài kiểm tra (0)
Điểm trung bình CBM	nan
Độ chính xác	nan%
Điểm thưởng CBM	nan%
Độ chính xác + thưởng	nan%
	Break-down by certainty
C=3	Không có trả lời
C=2	Không có trả lời
C=1	Không có trả lời

Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

In this exercise, you can use implemented functions in *previous question* (if needed) and implement these following functions.

1. Overload operator =
2. Overload operator == (The two circles are equal if they have the same center and radius)
3. Overload operator >> (stdin center.x, center.y, radius in order)

For example:

Test	Input	Result

`Point pointO(0, 0);`
`Circle A = Circle(pointO, 3);`
`Circle B;`
`B = A;`
`cout << (B == A);`
		`1`

`Circle A;`
`cin >> A;`
`A.printCircle();`
	`2`
	`3.5`
	`2`
		`Center: {2.00, 3.50} and Radius 2.00`

Answer: (penalty regime: 0 %)

Reset answer

```
class Point
{
    /*
     * STUDENT ANSWER
     * TODO: using code template in previous question
     */
private:
    double x,y;

public:
    Point() {
        this->x = 0;
        this->y = 0;
    }

    Point(double x, double y) {
        this->x = x;
        this->y = y;
    }

    double getX() {
        return this->x;
    }

    double getY() {
        return this->y;
    }

    void setX(double x) {
        this->x = x;
    }

    void setY(double y) {
        this->y = y;
    }

```

	Test	Input	Expected	Got	

✔	`Point pointO(0, 0);`
	`Circle A = Circle(pointO, 3);`
	`Circle B;`
	`B = A;`
	`cout << (B == A);`
			`1`	`1`	✔

✔	`Circle A;`
	`cin >> A;`
	`A.printCircle();`
		`2`
		`3.5`
		`2`
			`Center: {2.00, 3.50} and Radius 2.00`	`Center: {2.00, 3.50} and Radius 2.00`	✔

Passed all tests! ✔

Chính xác

Điểm cho bài nộp này: 1,00/1,00.

Câu hỏi 2

Không chính xác

Điểm 0,00 của 1,00

Given an array of integers.
Your task is to implement a function with following prototype:

```
int equalSumIndex(vector<int>& nums);
```

The function returns the smallest index `i` such that the sum of the numbers to the left of `i` is equal to the sum of the numbers to the right.
If no such index exists, return `-1`.

Note:
- The `iostream` and `vector` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result
`vector<int> nums {3, 5, 2, 7, 6, 4};` `cout << equalSumIndex(nums);`	3

Answer: (penalty regime: 0 %)

Reset answer

```
int equalSumIndex(vector<int>& nums) {
    // STUDENT ANSWER
}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int equalSumIndex(std::vector<int>&)':
__tester__.cpp:9:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 3

Không chính xác

Điểm 0,00 của 1,00

Given a string **s** containing just the characters **'('**, **')'**, **'['**, **']'**, **'{'**, **and '}'**. Check if the input string is valid based on following rules:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

For example:

- String **"[]()"** is a valid string, also **"[()]".**
- String "**[])**" is **not** a valid string.

Your task is to implement the function

```
bool isValidParentheses (string s){
    /*TODO*/
}
```

For example:

Test	Result
`cout << isValidParentheses("[]");`	`1`
`cout << isValidParentheses("[]()");`	`1`
`cout << isValidParentheses("[)");`	`0`

Answer: (penalty regime: 0 %)

Reset answer

```
bool isValidParentheses (string s){
    /*TODO*/
}
```

Syntax Error(s)

```
__tester__.cpp: In function 'bool isValidParentheses(std::__cxx11::string)':
__tester__.cpp:12:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 4

Không chính xác

Điểm 0,00 của 1,00

Given an array of positive integers `nums`. An array `mt` represents a mountain having no valleys and taking `nums` as its upperbound. In the other word, for all index `i` in range, `mt[i] ≤ nums[i]` and no pair of indices `(j, k)` that `j < i < k` and `mt[j] > mt[i] && mt[i] < mt[k]` exists.
Your task is to implement a function with following prototype:

```
int mountainWithoutValley(vector<int>& nums);
```

The function returns the maximum sum of numbers of `mt`.

Note:
- The `iostream`, `vector`, `climits` and `stack` libraries have been included and `namespace std` is being used. No other libraries are allowed.
- You can write helper functions.

For example:

Test	Result
`vector<int> nums {10, 6, 8, 8};` `cout << mountainWithoutValley(nums);`	28

Answer: (penalty regime: 0 %)

Reset answer

```
int mountainWithoutValley(vector<int>& nums) {
    // STUDENT ANSWER
}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int mountainWithoutValley(std::vector<int>&)':
__tester__.cpp:11:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 5

Đúng một phần

Điểm 0,10 của 1,00

In this question, you have to perform **add** on AVL tree. Note that:

- When adding a node which has the same value as parent node, add it in the **right sub tree.**

Your task is to implement function: **insert.** You could define one or more functions to achieve this task**.**

```
#include <iostream>
#include <math.h>
#include <queue>
using namespace std;
#define SEPARATOR "#<ab@17943918#@>#"

enum BalanceValue
{
    LH = -1,
    EH = 0,
    RH = 1
};

void printNSpace(int n)
{
    for (int i = 0; i < n - 1; i++)
        cout << " ";
}

void printInteger(int &n)
{
    cout << n << " ";
}

template<class T>
class AVLTree
{
public:
    class Node;
private:
    Node *root;
protected:
    int getHeightRec(Node *node)
    {
        if (node == NULL)
            return 0;
        int lh = this->getHeightRec(node->pLeft);
        int rh = this->getHeightRec(node->pRight);
        return (lh > rh ? lh : rh) + 1;
    }
public:
    AVLTree() : root(nullptr) {}
    ~AVLTree(){}
    int getHeight()
    {
        return this->getHeightRec(this->root);
    }
    void printTreeStructure()
    {
        int height = this->getHeight();
        if (this->root == NULL)
        {
            cout << "NULL\n";
            return;
        }
        queue<Node *> q;
        q.push(root);
        Node *temp;
        int count = 0;
        int maxNode = 1;
        int level = 0;
        int space = pow(2, height);
        printNSpace(space / 2);
        while (!q.empty())
        {
            temp = q.front();
            q.pop();
            if (temp == NULL)
            {
                cout << " ";
```

```
                q.push(NULL);
                q.push(NULL);
            }
            else
            {
                cout << temp->data;
                q.push(temp->pLeft);
                q.push(temp->pRight);
            }
            printNSpace(space);
            count++;
            if (count == maxNode)
            {
                cout << endl;
                count = 0;
                maxNode *= 2;
                level++;
                space /= 2;
                printNSpace(space / 2);
            }
            if (level == height)
                return;
        }
    }

    void insert(const T &value)
    {
        //TODO
    }

    class Node
    {
    private:
        T data;
        Node *pLeft, *pRight;
        BalanceValue balance;
        friend class AVLTree<T>;

    public:
        Node(T value) : data(value), pLeft(NULL), pRight(NULL), balance(EH) {}
        ~Node() {}
    };
};
```

For example:

Test	Result

`AVLTree<int> avl;`
`for (int i = 0; i < 9; i++){`
`  avl.insert(i);`
`}`
`avl.printTreeStructure();`
	`  3`
	`  1  5`
	`0  2  4  7`
	`  6 8`

`AVLTree<int> avl;`
`for (int i = 10; i >= 0; i--){`
`  avl.insert(i);`
`}`
`avl.printTreeStructure();`
	`  7`
	`  3  9`
	`1  5  8  10`
	`0 2 4 6`

Answer: (penalty regime: 0 %)

Reset answer

```
1  //Helping functions
2
3  void insert(const T &value){
4      //TODO
5
6  }
```

```
7
8
```

	Test	Expected	Got	

✖	`AVLTree<int> avl;`
	`for (int i = 0; i < 9; i++){`
	`  avl.insert(i);`
	`}`
	`avl.printTreeStructure();`
		`  3`
		`1  5`
		`0 2 4 7`
		`  6 8`
			NULL	✖

✖	`AVLTree<int> avl;`
	`for (int i = 10; i >= 0; i--){`
	`\tavl.insert(i);`
	`}`
	`avl.printTreeStructure();`
		`  7`
		`3  9`
		`1 5 8 10`
		`0 2 4 6`
			NULL	✖

Some hidden test cases failed, too.

Show differences

Đúng một phần

Điểm cho bài nộp này: 0,10/1,00.

Câu hỏi 6

Không trả lời

Điểm 0,00 của 1,00

In this question, you are to recreate an AVL tree in the previous insert question but the node will have 2 data instead of one and:

_ The two data have the same type.

_ Only the first data are used to calculate the insert process.

For example:

Test	Result

`AVLTree<int> avl;`
`for (int i = 0; i < 9; i++){`
`  avl.insert(i,8-i);`
`}`
`avl.printTreeStructure();`
	`  3, 5`
	`  1, 7  5, 3`
	`0, 8  2, 6  4, 4  7, 1`
	`  6, 2 8, 0`

`AVLTree<int> avl;`
`for (int i = 10; i >= 0; i--){`
`  avl.insert(i,i);`
`}`
`avl.printTreeStructure();`
	`  7, 7`
	`  3, 3  9, 9`
	`1, 1  5, 5  8, 8  10, 10`
	`0, 0 2, 2 4, 4 6, 6`

Answer: (penalty regime: 0 %)

```
1
```

Câu hỏi 7

Không chính xác

Điểm 0,00 của 1,00

Class **BTNode** is used to store a node in binary tree, described on the following:

```
class BTNode {
    public:
        int val;
        BTNode *left;
        BTNode *right;
        BTNode() {
            this->left = this->right = NULL;
        }
        BTNode(int val) {
            this->val = val;
            this->left = this->right = NULL;
        }
        BTNode(int val, BTNode*& left, BTNode*& right) {
            this->val = val;
            this->left = left;
            this->right = right;
        }
};
```

Where `val` is the value of node (non-negative integer), `left` and `right` are the pointers to the left node and right node of it, respectively.

Request: Implement function:

```
int secondDeepest(BTNode* root);
```

Where `root` is the root node of given binary tree (this tree has between 2 and 100000 elements). This function returns the depth of the second deepest leaf/leaves of the tree (if there is no leaf satisfying, return `-1`).

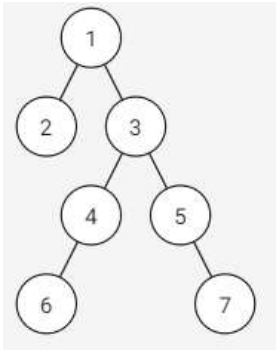
More information:

- The root has a depth of `0`.

- In a binary tree, the second deepest leaf's/leaves' depth is smaller than the deepest leaf/leaves's depth and higher than the others' depth.

Example:

Given a binary tree in the following:



The second deepest leaf is node `2`, the depth of node `2` is `1`; therefore, the function returns `1`.

Note: In this exercise, the libraries `iostream`, `stack`, `queue`, `utility` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {-1,0,0,2,2,3,4};`
`int value[] = {1,2,3,4,5,6,7};`
`BTNode* root = BTNode::createTree(arr, arr + sizeof(arr)/sizeof(int), value);`
`cout << secondDeepest(root);`
	1

`int arr[] = {-1,0,1,2,3,4,5,6,7,8};`
`int value[] = {1,2,3,4,5,6,7,8,9,10};`
`BTNode* root = BTNode::createTree(arr, arr + sizeof(arr)/sizeof(int), value);`
`cout << secondDeepest(root);`
	-1

Answer: (penalty regime: 0 %)

Reset answer

```
int secondDeepest(BTNode* root) {

}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int secondDeepest(BTNode*)':
__tester__.cpp:103:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 8

Không chính xác

Điểm 0,00 của 1,00

Class **BTNode** is used to store a node in binary search tree, described on the following:

```
class BTNode {
    public:
        int val;
        BTNode *left;
        BTNode *right;
        BTNode() {
            this->left = this->right = NULL;
        }
        BTNode(int val) {
            this->val = val;
            this->left = this->right = NULL;
        }
        BTNode(int val, BTNode*& left, BTNode*& right) {
            this->val = val;
            this->left = left;
            this->right = right;
        }
};
```

Where `val` is the value of node (non-negative integer), `left` and `right` are the pointers to the left node and right node of it, respectively.

Request: Implement function:

```
BTNode* enlarge(BTNode* root);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements), return the tree after **enlarging**.

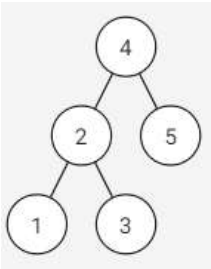
More information:

- There are no two nodes with the same `val` in this binary search tree.

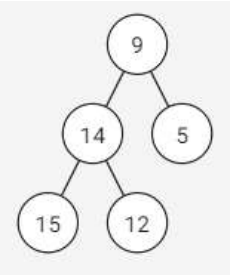
- **Enlarging** a binary search tree is replacing each node's `val` of this tree by sum of its `val` and all other `val`s in the tree that are greater than its `val`.

Example:

Given a binary search tree in the following:



The tree after enlarging is:



Note: In this exercise, the libraries `iostream`, `stack`, `queue`, `utility` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {4,2,5,1,3};`
`BTNode* root= BTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`root = enlarge(root);`
`BTNode::printInorder(root);`
	`15 14 12 9 5`

`int arr[] = {2,1,0};`
`BTNode* root= BTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`root = enlarge(root);`
`BTNode::printInorder(root);`
	`3 3 2`

Answer: (penalty regime: 0 %)

Reset answer

```
BTNode* enlarge(BTNode* root) {

}
```

Syntax Error(s)

```
__tester__.cpp: In function 'BTNode* enlarge(BTNode*)':
__tester__.cpp:103:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 9

Không chính xác

Điểm 0,00 của 1,00

Implement function to detect a cyclic in Graph

```
bool isCyclic();
```

Graph structure in this lab is slightly different from previous labs.

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

class DirectedGraph
{
        int V;
        list<int> *adj;
        bool isCyclicUtil(int v, bool visited[], bool *rs);
public:
        DirectedGraph(){
        V = 0;
        adj = NULL;
    }
        DirectedGraph(int V)
        {
                this->V = V;
                adj = new list<int>[V];
        }
        void addEdge(int v, int w)
        {
                adj[v].push_back(w);
        }
        bool isCyclic();
};
```

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 <list>
using namespace std;

class DirectedGraph
{
    int V;
    list<int> *adj;
```

```
    bool isCyclicUtil(int v, bool visited[], bool *rs);
public:
    DirectedGraph(){
        V = 0;
        adj = NULL;
    }
    DirectedGraph(int V)
    {
        this->V = V;
        adj = new list<int>[V];
    }
    void addEdge(int v, int w)
    {
        adj[v].push_back(w);
    }
    bool isCyclic()
    {
        // Student answer
    }
};
```

Syntax Error(s)

```
__tester__.cpp: In member function 'bool DirectedGraph::isCyclic()':
__tester__.cpp:37:2: error: no return statement in function returning non-void [-Werror=return-type]
  }
  ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 10

Không chính xác

Điểm 0,00 của 1,00

Given a sequence of sorted integers whose prime factors only include `2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31` and starts at `1`.

Request: Implement function:

```
int uglyNumberIndex(int n);
```

This function returns the `n`-th indexed element of the sequence. (The sequence is `0`-th indexed, and `n` is in range `[0, 100000]`).

Example:

The first elements of the sequence: `[1, 2, 3, 4, 5, 6, 7,...]`

The `2`-nd indexed element is `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 class. Importing other libraries is allowed, but not encouraged.

For example:

Test	Result
`int n = 5;` `cout << uglyNumberIndex(n);`	6

Answer: (penalty regime: 10, 20, ... %)

Reset answer

```
int uglyNumberIndex(int n) {
    // STUDENT ANSWER
}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int uglyNumberIndex(int)':
__tester__.cpp:22:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 11

Không chính xác

Điểm 0,00 của 1,00

Implement function

```
int interpolationSearch(int arr[], int left, int right, int x)
```

to search for value x in array arr using recursion.
After traverse to an index in array, before returning the index or passing it as argument to recursive function, we print out this index using cout << "We traverse on index: " << index << endl;

Please note that you can't using key work for, while, goto (even in variable names, comment).

For example:

Test	Result

`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int x = 3;`
`int result = interpolationSearch(arr, 0, n - 1, x);`
`(result == -1) ? cout << "Element is not present in array"`
`: cout << "Element is present at index " << result;`
	`We traverse on index: 2`
	`Element is present at index 2`

`int arr[] = { 1,2,3,4,5,6,7,8,9 };`
`int n = sizeof(arr) / sizeof(arr[0]);`
`int x = 0;`
`int result = interpolationSearch(arr, 0, n - 1, x);`
`(result == -1) ? cout << "Element is not present in array"`
`: cout << "Element is present at index " << result;`
	`Element is not present in array`

Answer: (penalty regime: 0 %)

Reset answer

```
int interpolationSearch(int arr[], int left, int right, int x)
{

}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int interpolationSearch(int*, int, int, int)':
__tester__.cpp:9:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.

Câu hỏi 12

Không chính xác

Điểm 0,00 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) {

}
```

Syntax Error(s)

```
__tester__.cpp: In function 'int pairMatching(std::vector<int>&, int)':
__tester__.cpp:22:1: error: no return statement in function returning non-void [-Werror=return-type]
 }
 ^
cc1plus: all warnings being treated as errors
```

Không chính xác

Điểm cho bài nộp này: 0,00/1,00.