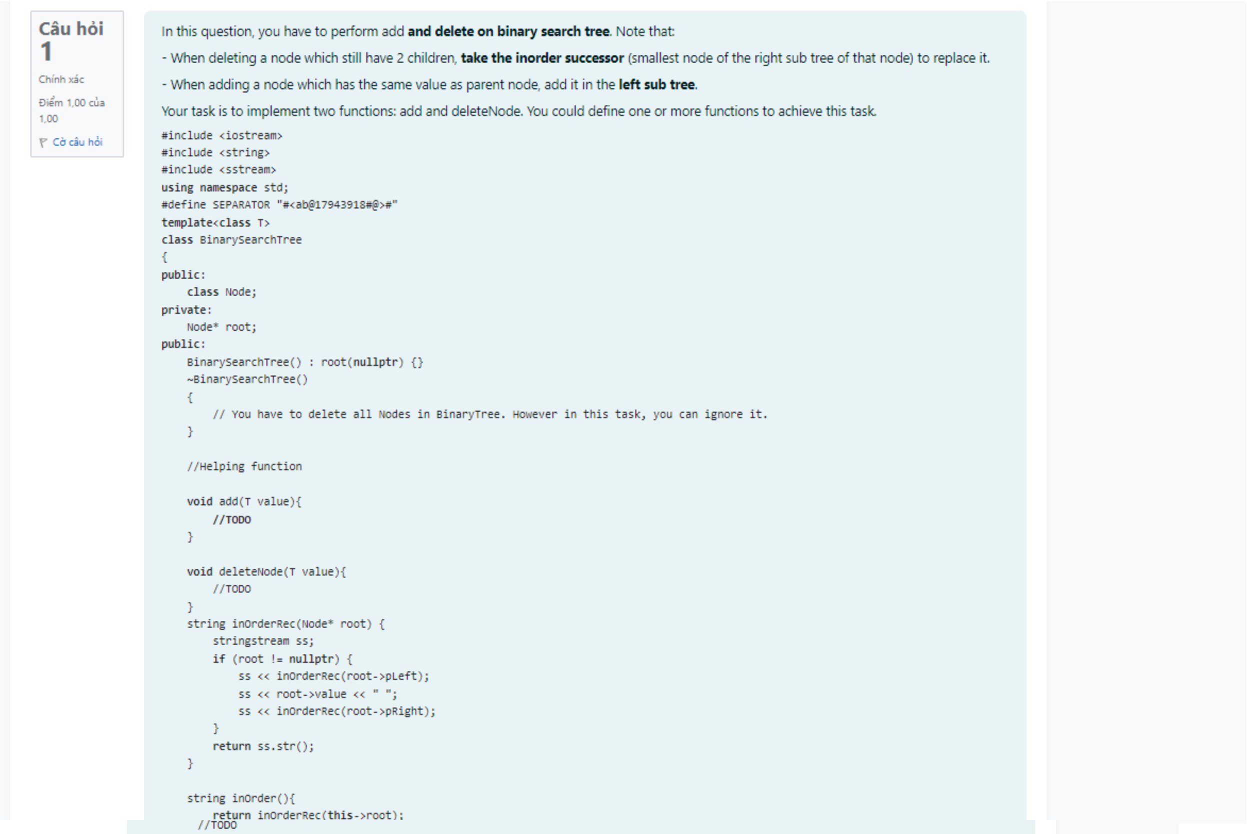


Câu hỏi 1

Chính xác

Điểm 1,00 của 1,00

Cờ câu hỏi

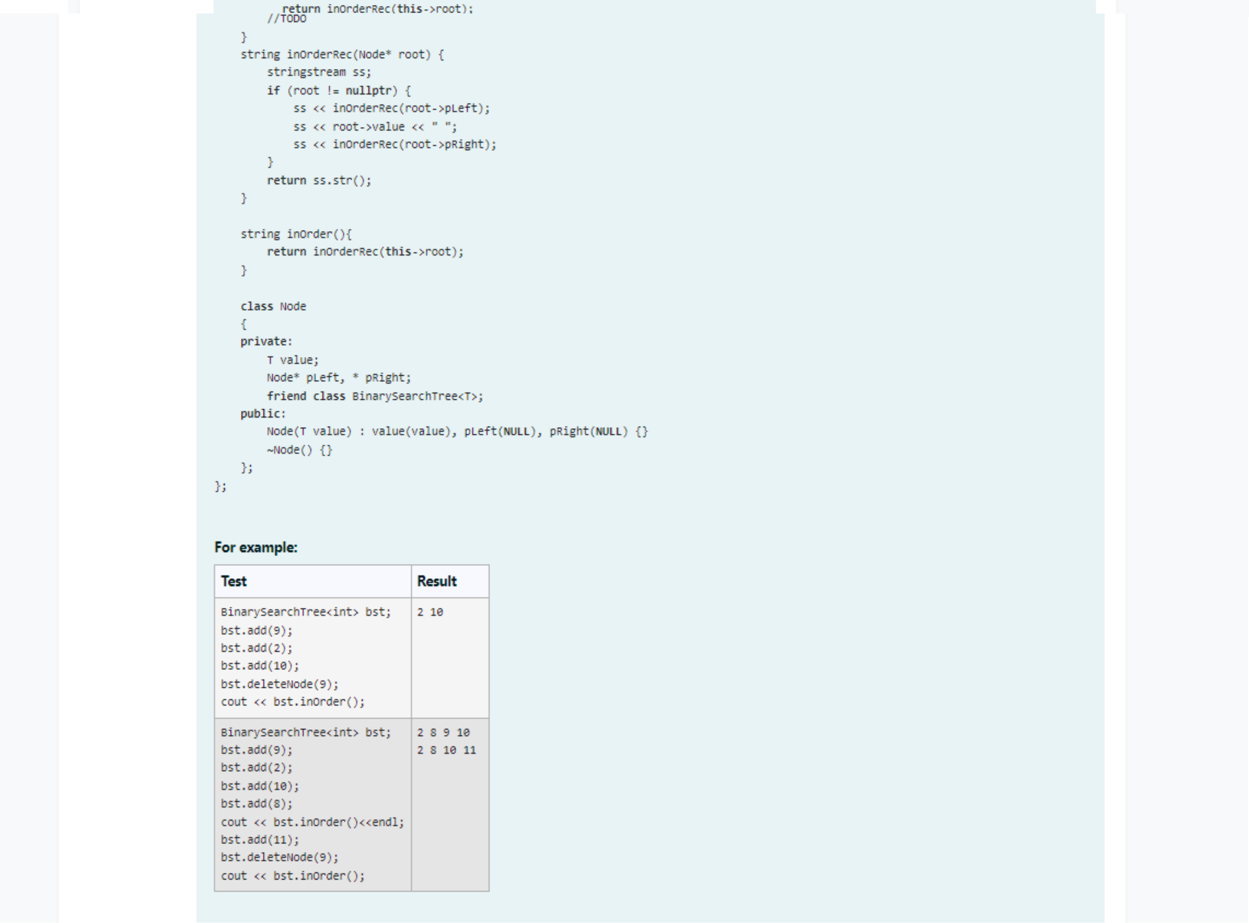
In this question, you have to perform add **and delete on binary search tree**. Note that:

- When deleting a node which still have 2 children, **take the inorder successor** (smallest node of the right sub tree of that node) to replace it.

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

Your task is to implement two functions: add and deleteNode. You could define one or more functions to achieve this task.

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
#define SEPARATOR "#<ab@17943918#@>#"
template<class T>
class BinarySearchTree
{
public:
    class Node;
private:
    Node* root;
public:
    BinarySearchTree() : root(nullptr) {}
    ~BinarySearchTree()
    {
        // You have to delete all Nodes in BinaryTree. However in this task, you can ignore it.
    }

    //Helping function

    void add(T value){
        //TODO
    }

    void deleteNode(T value){
        //TODO
    }
    string inOrderRec(Node* root) {
        stringstream ss;
        if (root != nullptr) {
            ss << inOrderRec(root->pLeft);
            ss << root->value << " ";
            ss << inOrderRec(root->pRight);
        }
        return ss.str();
    }

    string inOrder(){
        return inOrderRec(this->root);
```



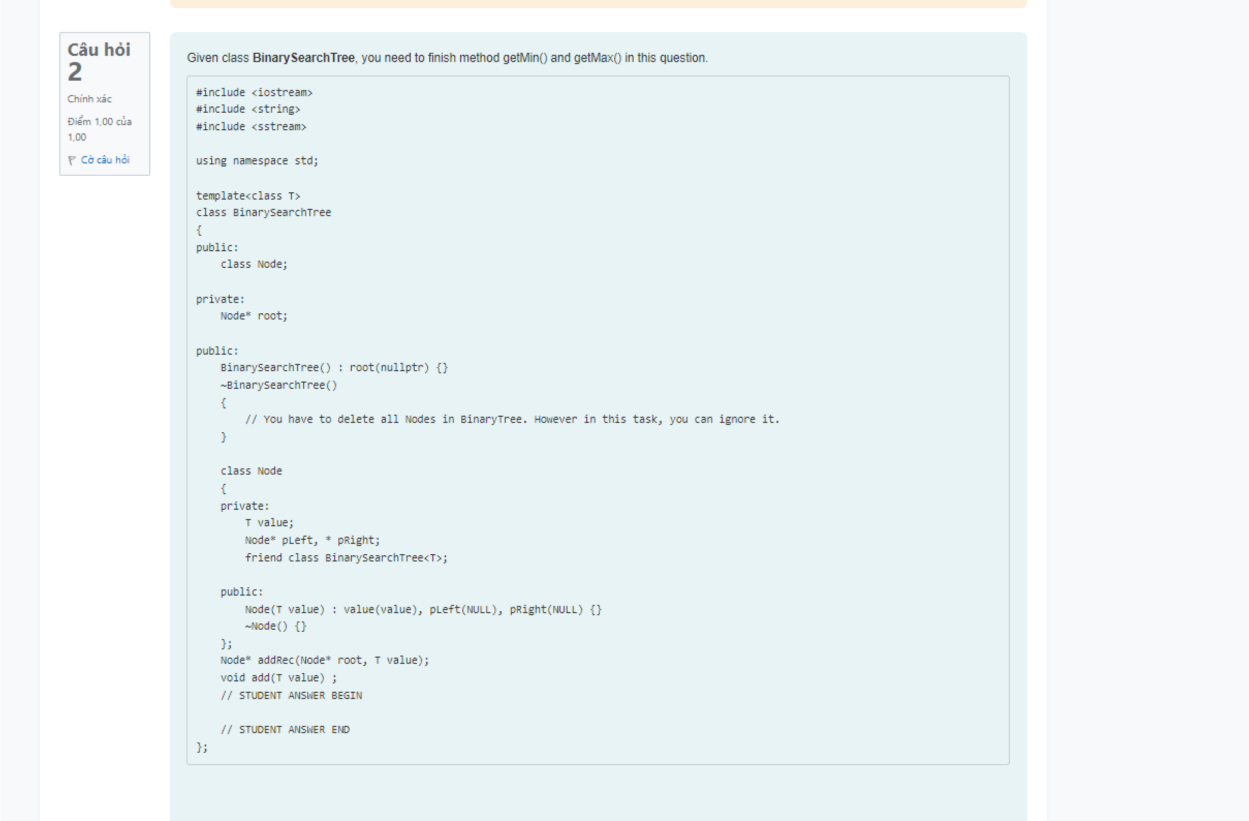
```
        //TODO
    }
    string inOrderRec(Node* root) {
        stringstream ss;
        if (root != nullptr) {
            ss << inOrderRec(root->pLeft);
            ss << root->value << " ";
            ss << inOrderRec(root->pRight);
        }
        return ss.str();
    }

    string inOrder(){
        return inOrderRec(this->root);
    }

    class Node
    {
    private:
        T value;
        Node* pLeft, * pRight;
        friend class BinarySearchTree<T>;
    public:
        Node(T value) : value(value), pLeft(NULL), pRight(NULL) {}
        ~Node() {}
    };
};
```

For example:

Test	Result

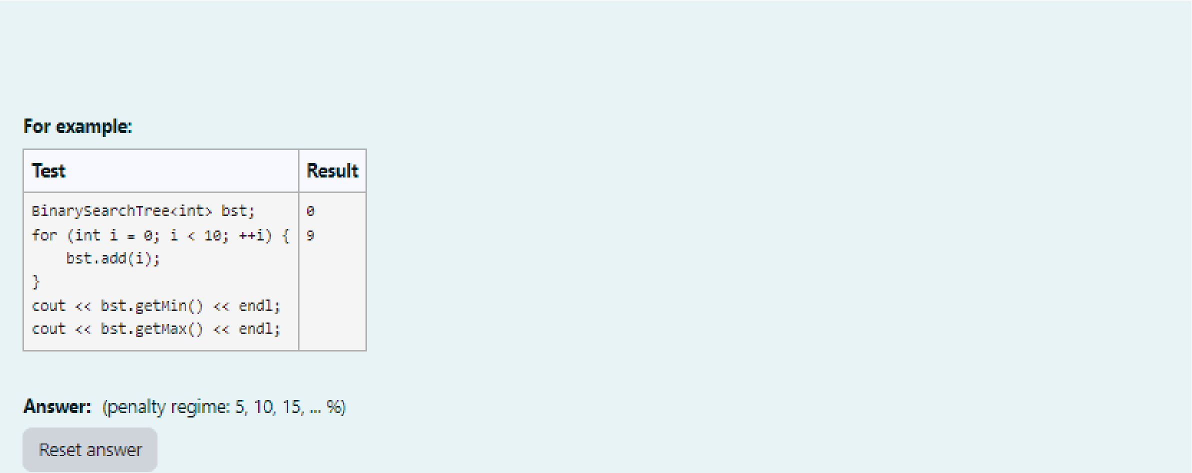
`BinarySearchTree<int> bst;`
`bst.add(9);`
`bst.add(2);`
`bst.add(10);`
`bst.deleteNode(9);`
`cout << bst.inOrder();`
	`2 10`

`BinarySearchTree<int> bst;`
`bst.add(9);`
`bst.add(2);`
`bst.add(10);`
`bst.add(8);`
`cout << bst.inOrder()<<endl;`
`bst.add(11);`
`bst.deleteNode(9);`
`cout << bst.inOrder();`
	`2 8 9 10`
	`2 8 10 11`



Given class **BinarySearchTree**, you need to finish method getMin() and getMax() in this question.

```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

template<class T>
class BinarySearchTree
{
public:
    class Node;

private:
    Node* root;

public:
    BinarySearchTree() : root(nullptr) {}
    ~BinarySearchTree()
    {
        // You have to delete all Nodes in BinaryTree. However in this task, you can ignore it.
    }

    class Node
    {
    private:
        T value;
        Node* pLeft, * pRight;
        friend class BinarySearchTree<T>;

    public:
        Node(T value) : value(value), pLeft(NULL), pRight(NULL) {}
        ~Node() {}
    };
    Node* addRec(Node* root, T value);
    void add(T value) ;
    // STUDENT ANSWER BEGIN

    // STUDENT ANSWER END
};
```

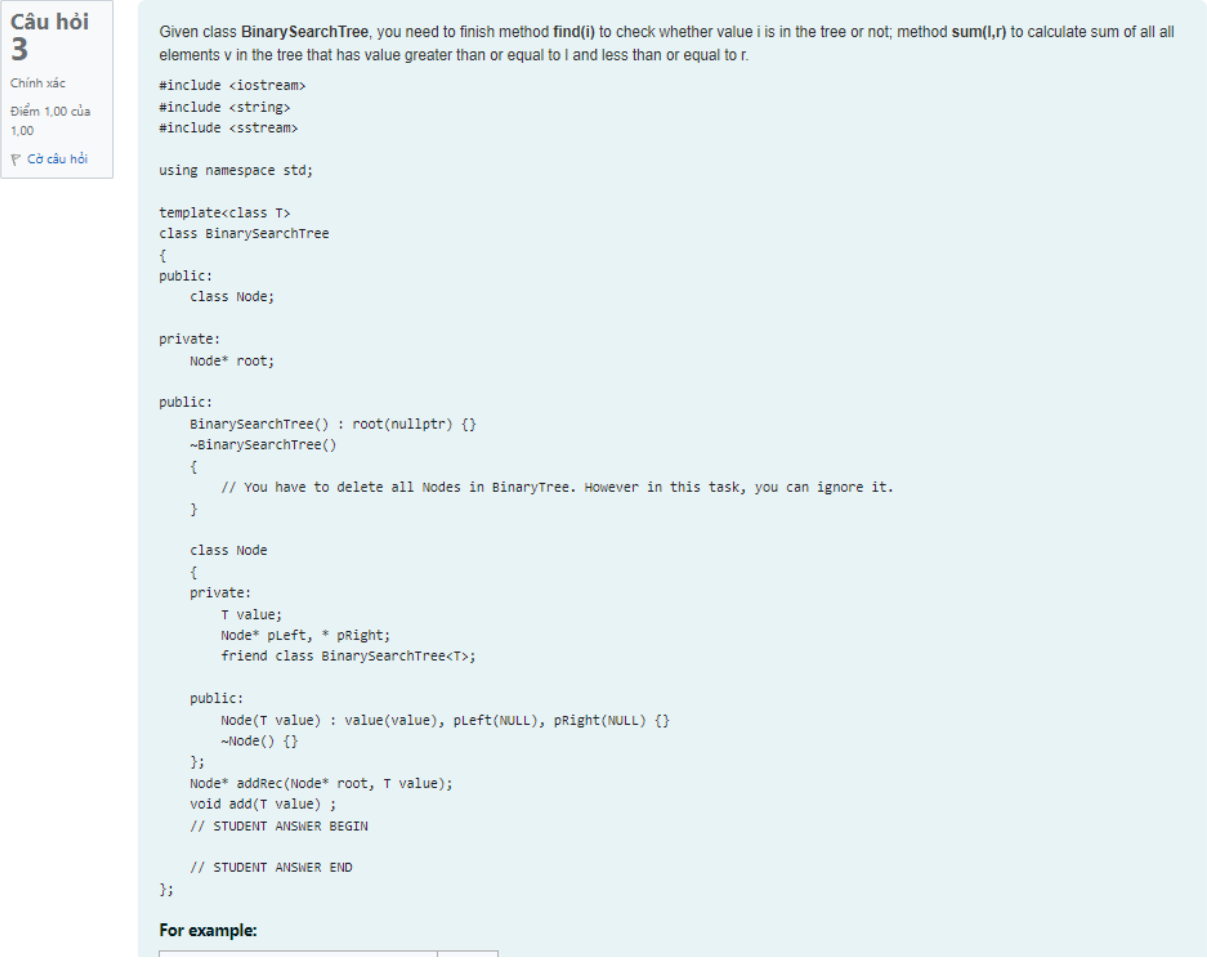


For example:

Test	Result

`BinarySearchTree<int> bst;`
`for (int i = 0; i < 10; ++i) {`
`    bst.add(i);`
`}`
`cout << bst.getMin() << endl;`
`cout << bst.getMax() << endl;`
	0
	9

Answer: (penalty regime: 5, 10, 15, ... %)



Câu hỏi **3**

Chính xác

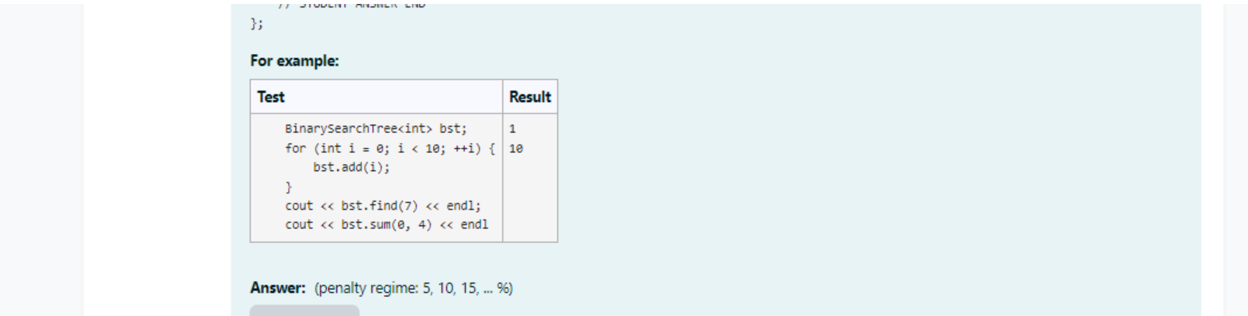
Điểm 1,00 của 1,00

Cờ câu hỏi

Given class **BinarySearchTree**, you need to finish method **find(i)** to check whether value i is in the tree or not; method **sum(l,r)** to calculate sum of all all elements v in the tree that has value greater than or equal to l and less than or equal to r.

```
#include <iostream>
#include <string>
#include <sstream>

using namespace std;

template<class T>
class BinarySearchTree
{
public:
    class Node;

private:
    Node* root;

public:
    BinarySearchTree() : root(nullptr) {}
    ~BinarySearchTree()
    {
        // You have to delete all Nodes in BinaryTree. However in this task, you can ignore it.
    }

    class Node
    {
    private:
        T value;
        Node* pLeft, * pRight;
        friend class BinarySearchTree<T>;

    public:
        Node(T value) : value(value), pLeft(NULL), pRight(NULL) {}
        ~Node() {}
    };
    Node* addRec(Node* root, T value);
    void add(T value) ;
    // STUDENT ANSWER BEGIN

    // STUDENT ANSWER END
};
```

For example:



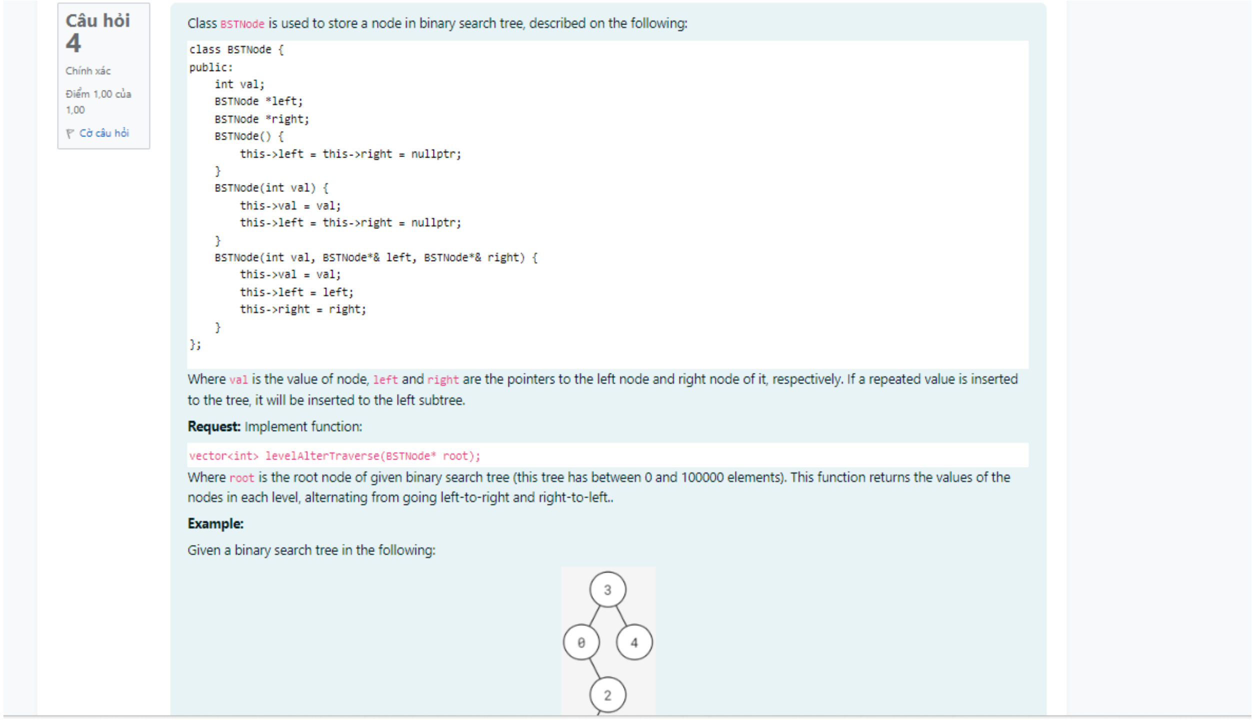
// STUDENT ANSWER END
};

For example:

Test	Result

BinarySearchTree<int> bst;
for (int i = 0; i < 10; ++i) {
    bst.add(i);
}
cout << bst.find(7) << endl;
cout << bst.sum(0, 4) << endl
	1
	10

Answer: (penalty regime: 5, 10, 15, ... %)



Câu hỏi 4

Chính xác

Điểm 1,00 của 1,00

Cờ câu hỏi

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

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

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

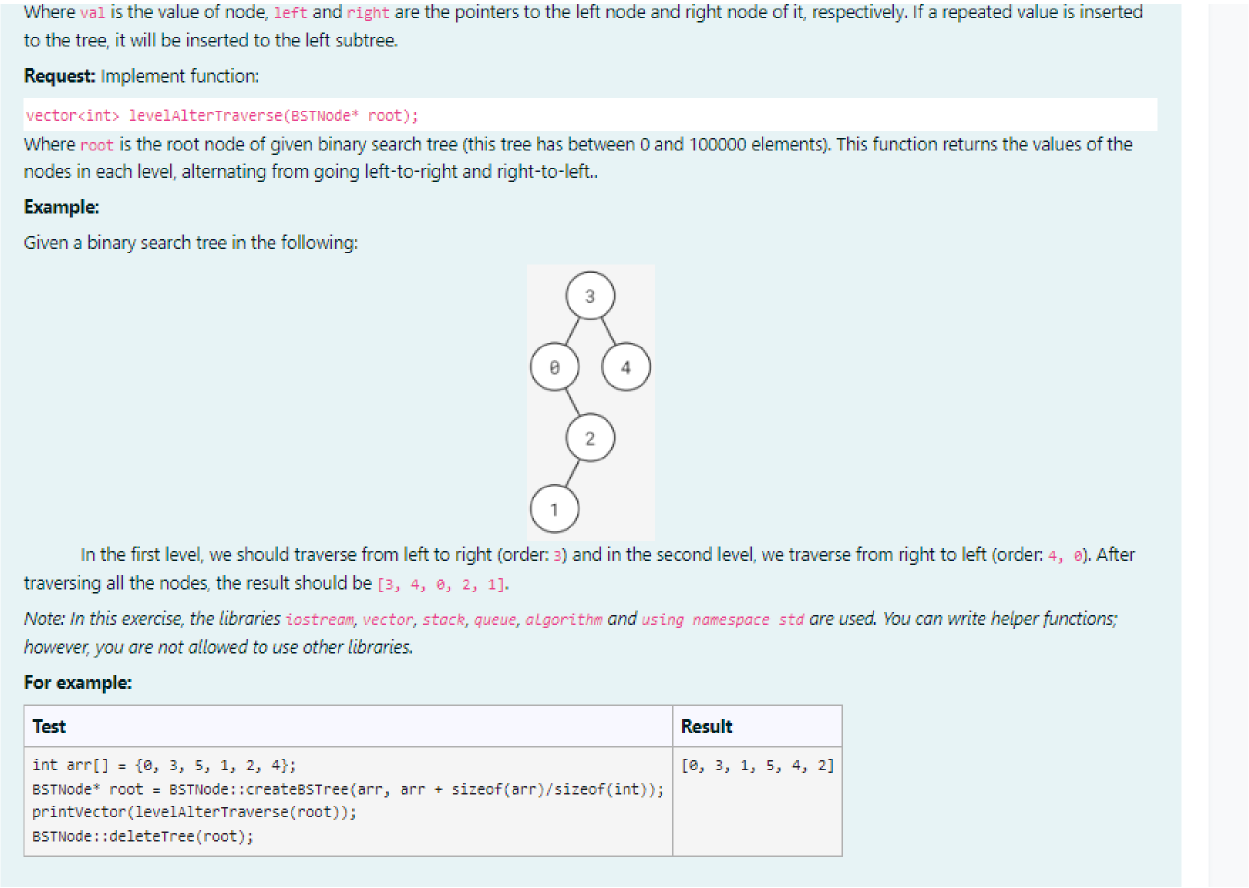
Request: Implement function:

```
vector<int> levelAlterTraverse(BSTNode* root);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the values of the nodes in each level, alternating from going left-to-right and right-to-left..

Example:

Given a binary search tree in the following:



Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Request: Implement function:

```
vector<int> levelAlterTraverse(BSTNode* root);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the values of the nodes in each level, alternating from going left-to-right and right-to-left..

Example:

Given a binary search tree in the following:

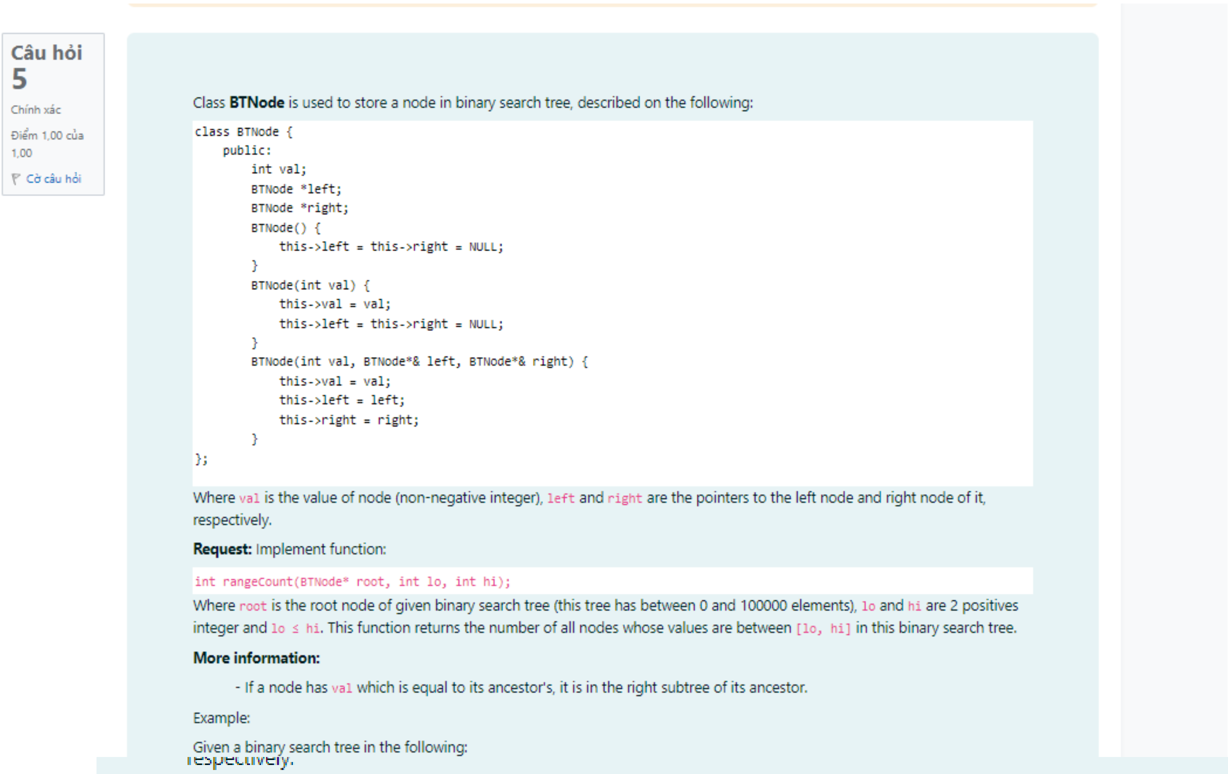
In the first level, we should traverse from left to right (order: `3`) and in the second level, we traverse from right to left (order: `4, 0`). After traversing all the nodes, the result should be `[3, 4, 0, 2, 1]`.

Note: In this exercise, the libraries `iostream`, `vector`, `stack`, `queue`, `algorithm` 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[] = {0, 3, 5, 1, 2, 4};`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`printVector(levelAlterTraverse(root));`
`BSTNode::deleteTree(root);`
	`[0, 3, 1, 5, 4, 2]`



Câu hỏi **5**

Chính xác

Điểm 1,00 của 1,00

Cờ câu hỏi

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:

```
int rangeCount(BTNode* root, int lo, int hi);
```

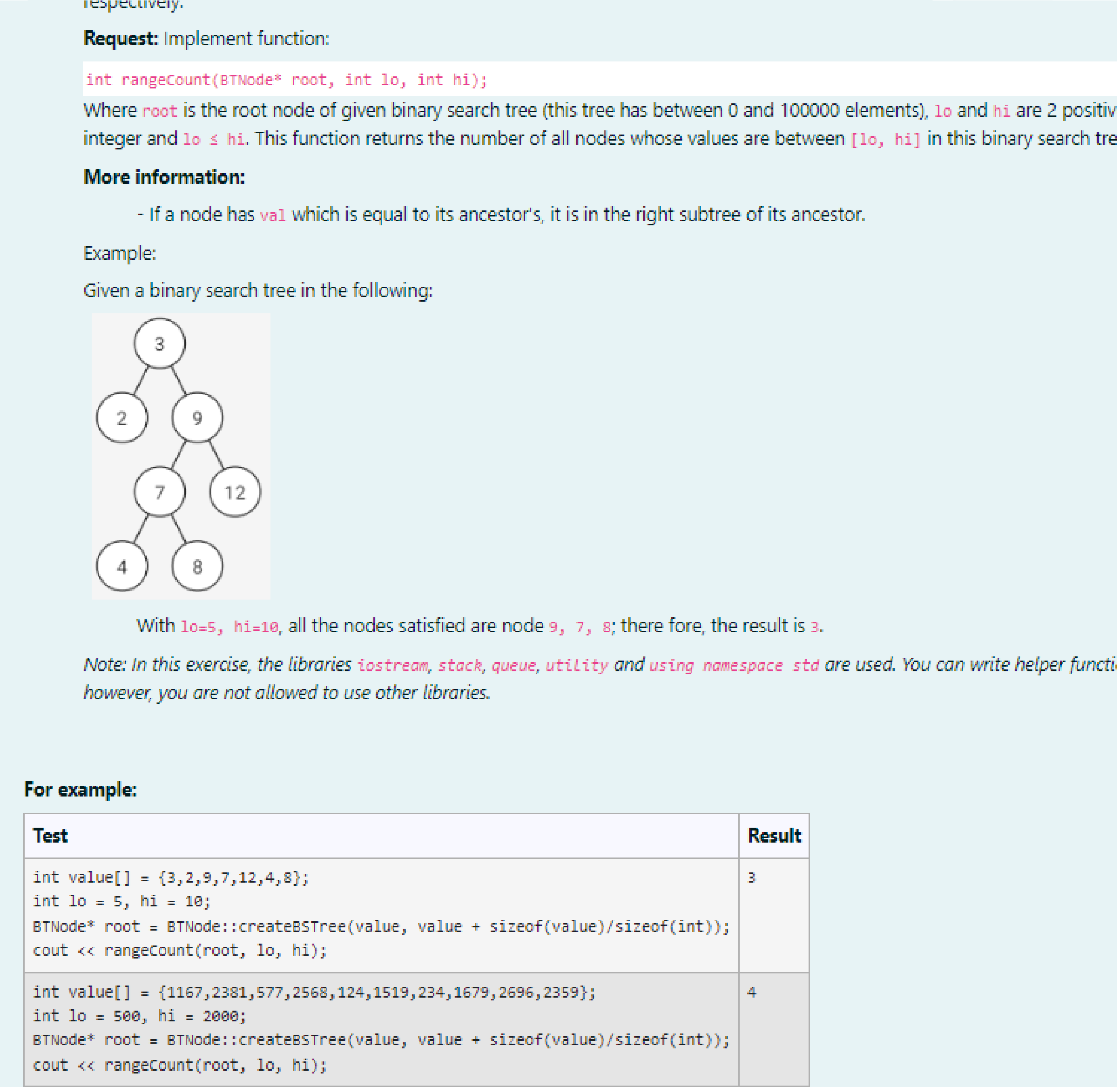
Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements), `lo` and `hi` are 2 positives integer and `lo ≤ hi`. This function returns the number of all nodes whose values are between `[lo, hi]` in this binary search tree.

More information:

- If a node has `val` which is equal to its ancestor's, it is in the right subtree of its ancestor.

Example:

Given a binary search tree in the following:



respectively.

Request: Implement function:

```
int rangeCount(BTNode* root, int lo, int hi);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements), `lo` and `hi` are 2 positiv integer and `lo ≤ hi`. This function returns the number of all nodes whose values are between `[lo, hi]` in this binary search tre

More information:

- If a node has `val` which is equal to its ancestor's, it is in the right subtree of its ancestor.

Example:

Given a binary search tree in the following:

With `lo=5, hi=10`, all the nodes satisfied are node `9, 7, 8`; there fore, the result is `3`.

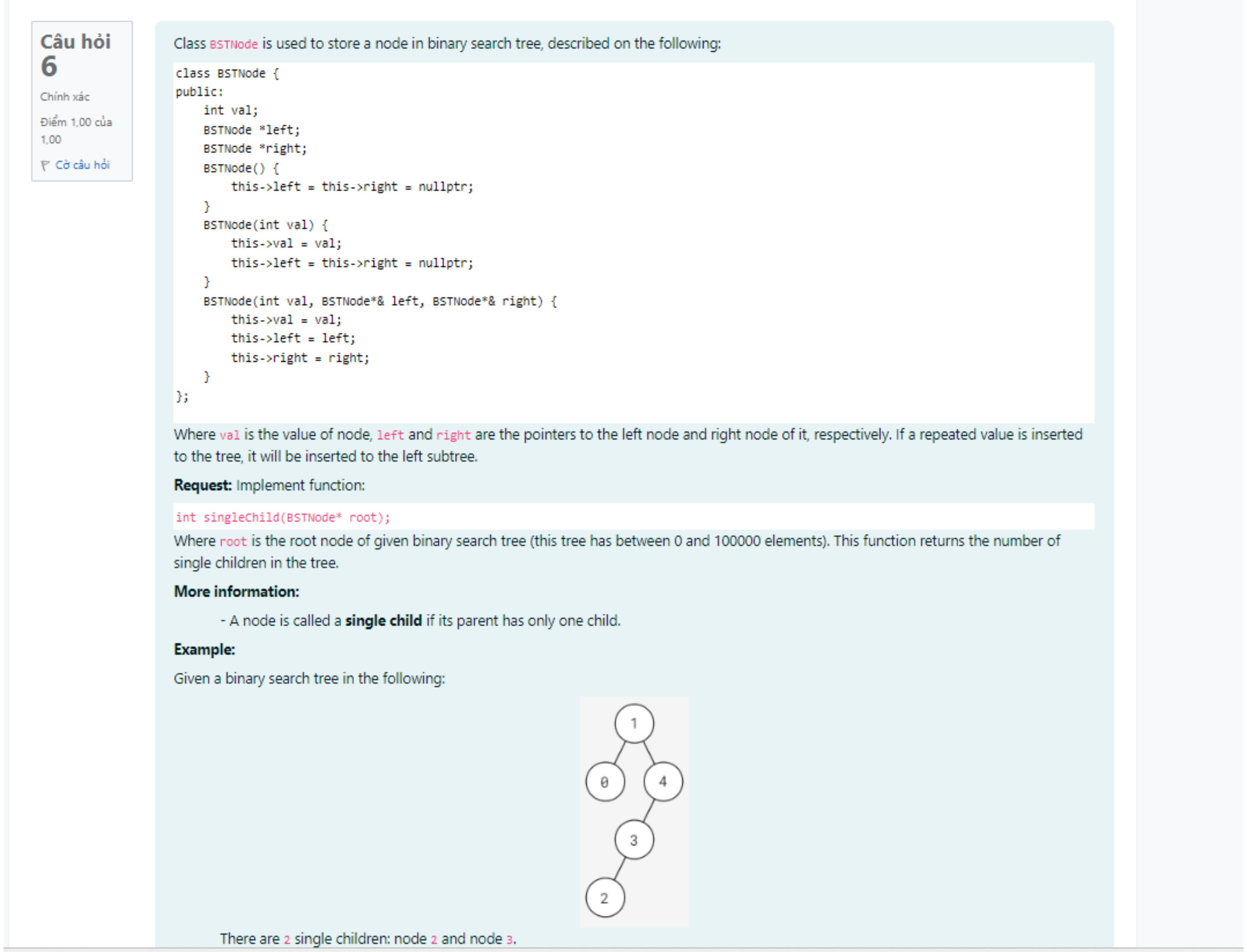
Note: In this exercise, the libraries `iostream`, `stack`, `queue`, `utility` and `using namespace std` are used. You can write helper functi however, you are not allowed to use other libraries.

For example:

Test	Result

`int value[] = {3,2,9,7,12,4,8};`
`int lo = 5, hi = 10;`
`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
`cout << rangeCount(root, lo, hi);`
	3

`int value[] = {1167,2381,577,2568,124,1519,234,1679,2696,2359};`
`int lo = 500, hi = 2000;`
`BTNode* root = BTNode::createBSTree(value, value + sizeof(value)/sizeof(int));`
`cout << rangeCount(root, lo, hi);`
	4



Câu hỏi **6**

Chính xác

Điểm 1,00 của 1,00

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

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

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Request: Implement function:

```
int singleChild(BSTNode* root);
```

Where `root` is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the number of single children in the tree.

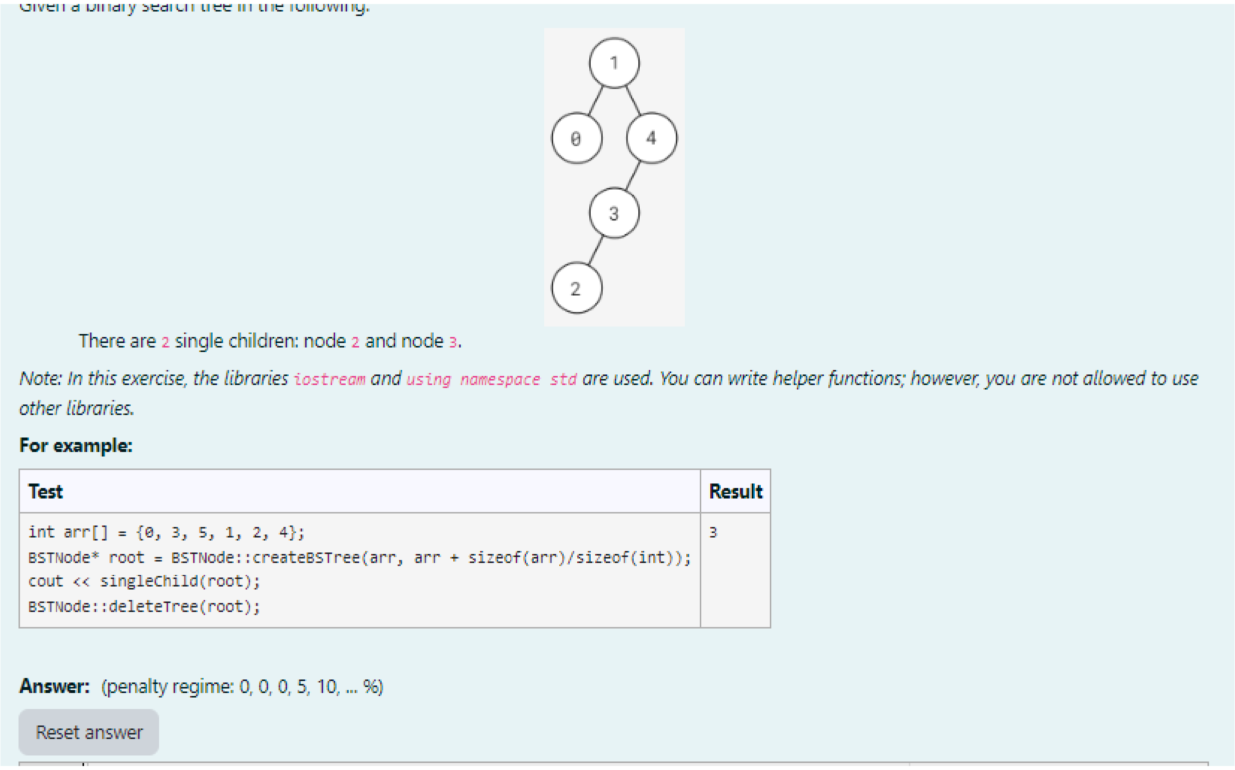
More information:

- A node is called a **single child** if its parent has only one child.

Example:

Given a binary search tree in the following:

There are `2` single children: node `2` and node `3`.



Given a binary search tree in the following:

There are `2` single children: node `2` and node `3`.

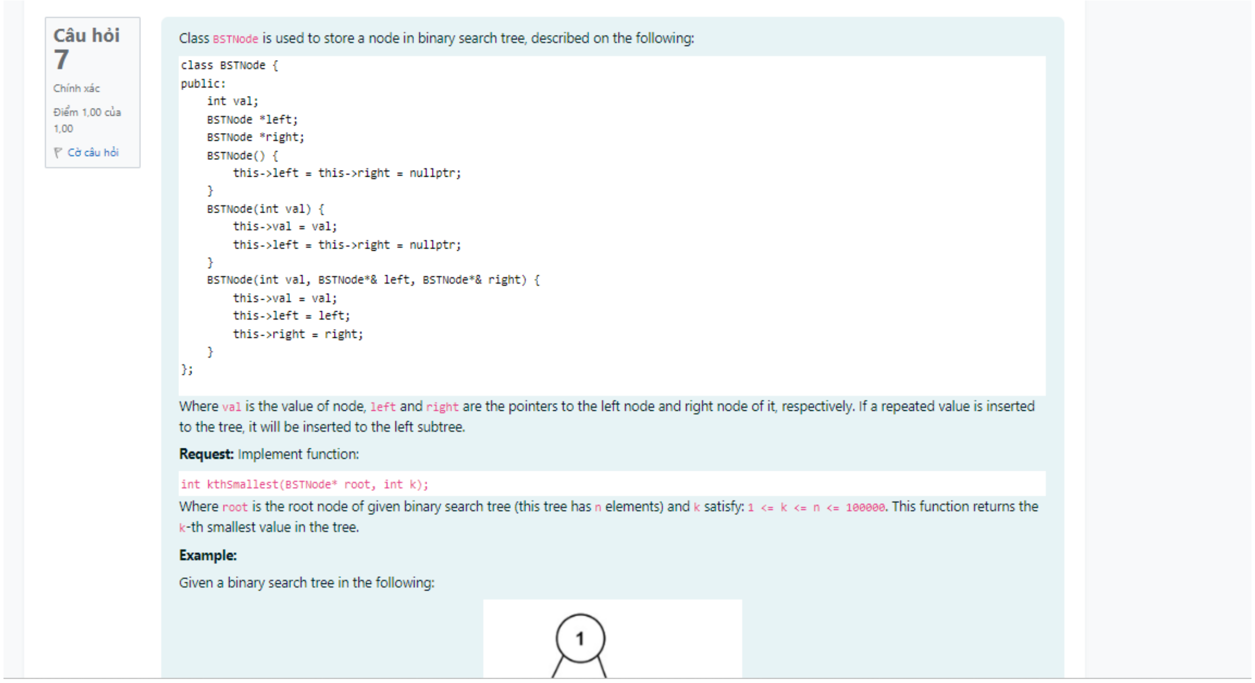
Note: In this exercise, the libraries `iostream` and `using namespace std` are used. You can write helper functions; however, you are not allowed to use other libraries.

For example:

Test	Result

`int arr[] = {0, 3, 5, 1, 2, 4};`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`cout << singleChild(root);`
`BSTNode::deleteTree(root);`
	3

Answer: (penalty regime: 0, 0, 0, 5, 10, ... %)



Câu hỏi **7**

Chính xác

Điểm 1,00 của 1,00

Cờ câu hỏi

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

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

Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

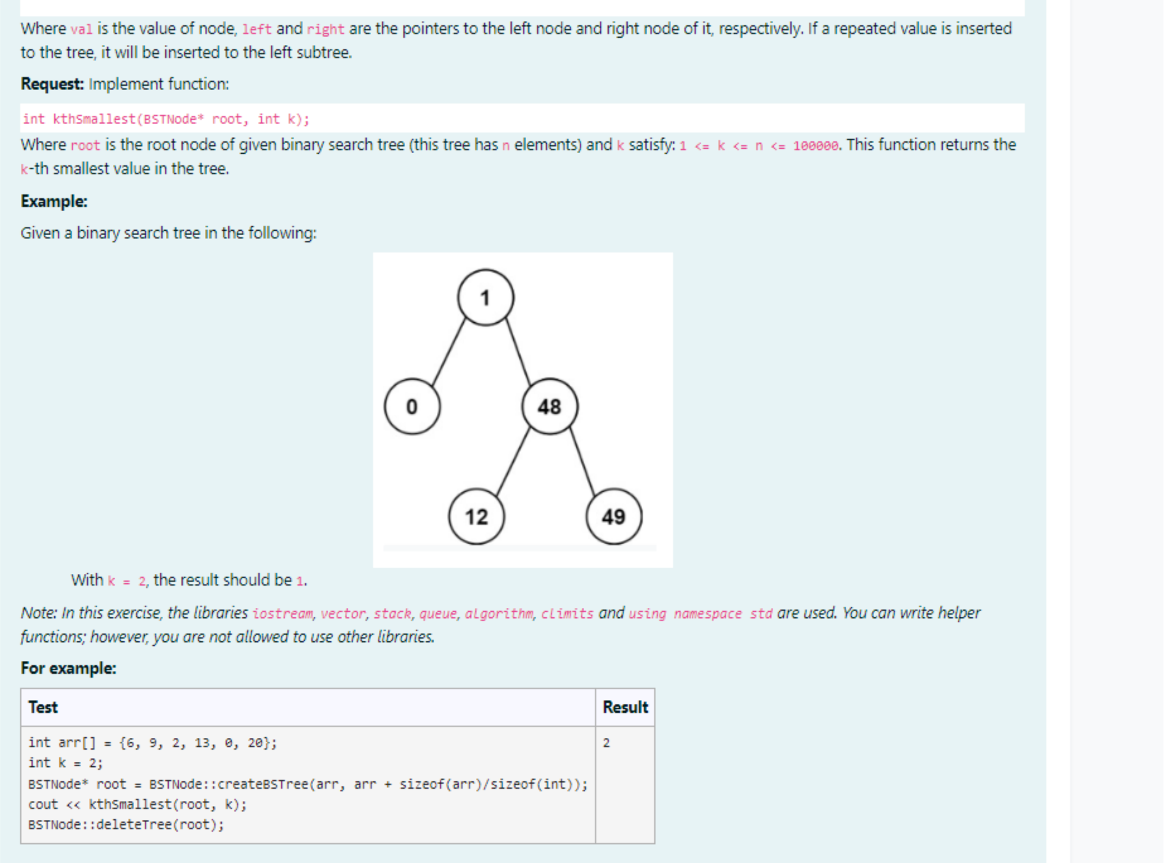
Request: Implement function:

`int kthSmallest(BSTNode* root, int k);`

Where `root` is the root node of given binary search tree (this tree has `n` elements) and `k` satisfy: `1 <= k <= n <= 100000`. This function returns the `k`-th smallest value in the tree.

Example:

Given a binary search tree in the following:



Where `val` is the value of node, `left` and `right` are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Request: Implement function:

`int kthSmallest(BSTNode* root, int k);`

Where `root` is the root node of given binary search tree (this tree has `n` elements) and `k` satisfy: `1 <= k <= n <= 100000`. This function returns the `k`-th smallest value in the tree.

Example:

Given a binary search tree in the following:

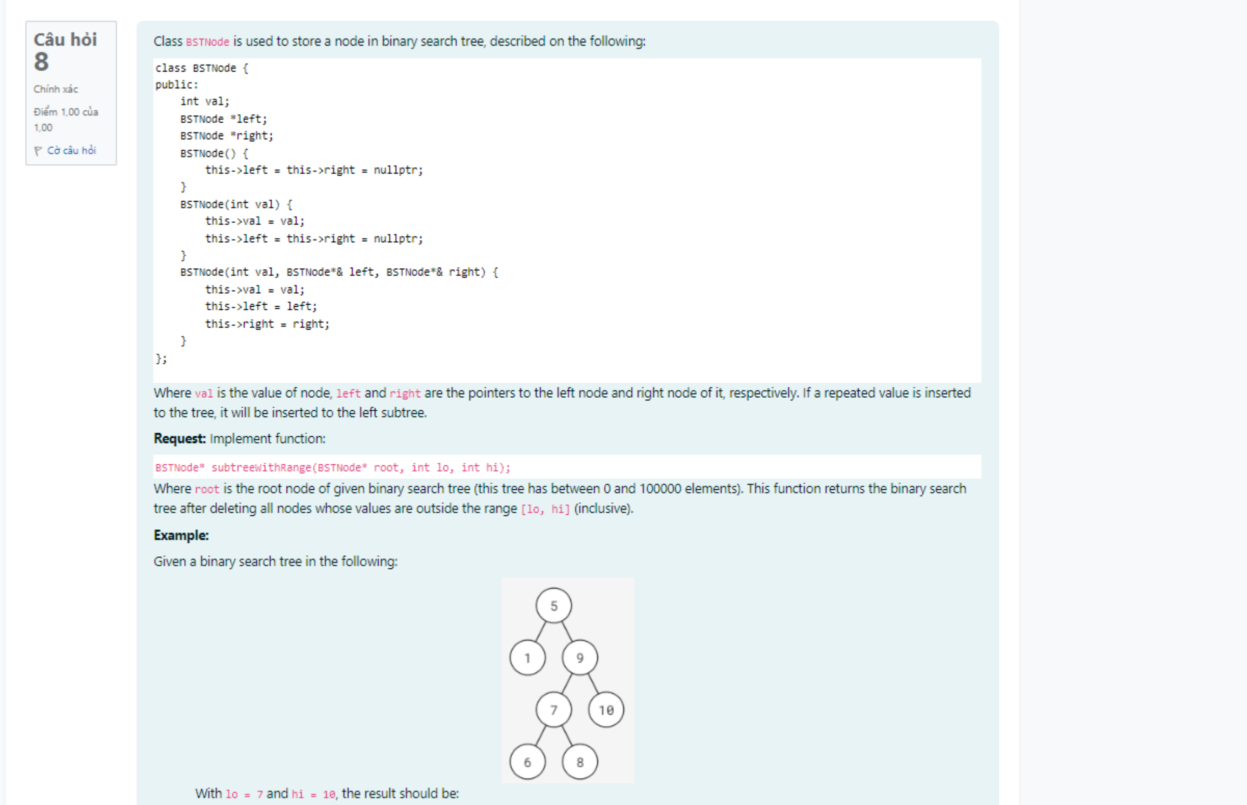
With `k = 2`, the result should be `1`.

Note: In this exercise, the libraries `iostream`, `vector`, `stack`, `queue`, `algorithm`, `climits` 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[] = {6, 9, 2, 13, 0, 20};`
`int k = 2;`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`cout << kthSmallest(root, k);`
`BSTNode::deleteTree(root);`
	2



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

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

Where val is the value of node, left and right are the pointers to the left node and right node of it, respectively. If a repeated value is inserted to the tree, it will be inserted to the left subtree.

Request: Implement function:

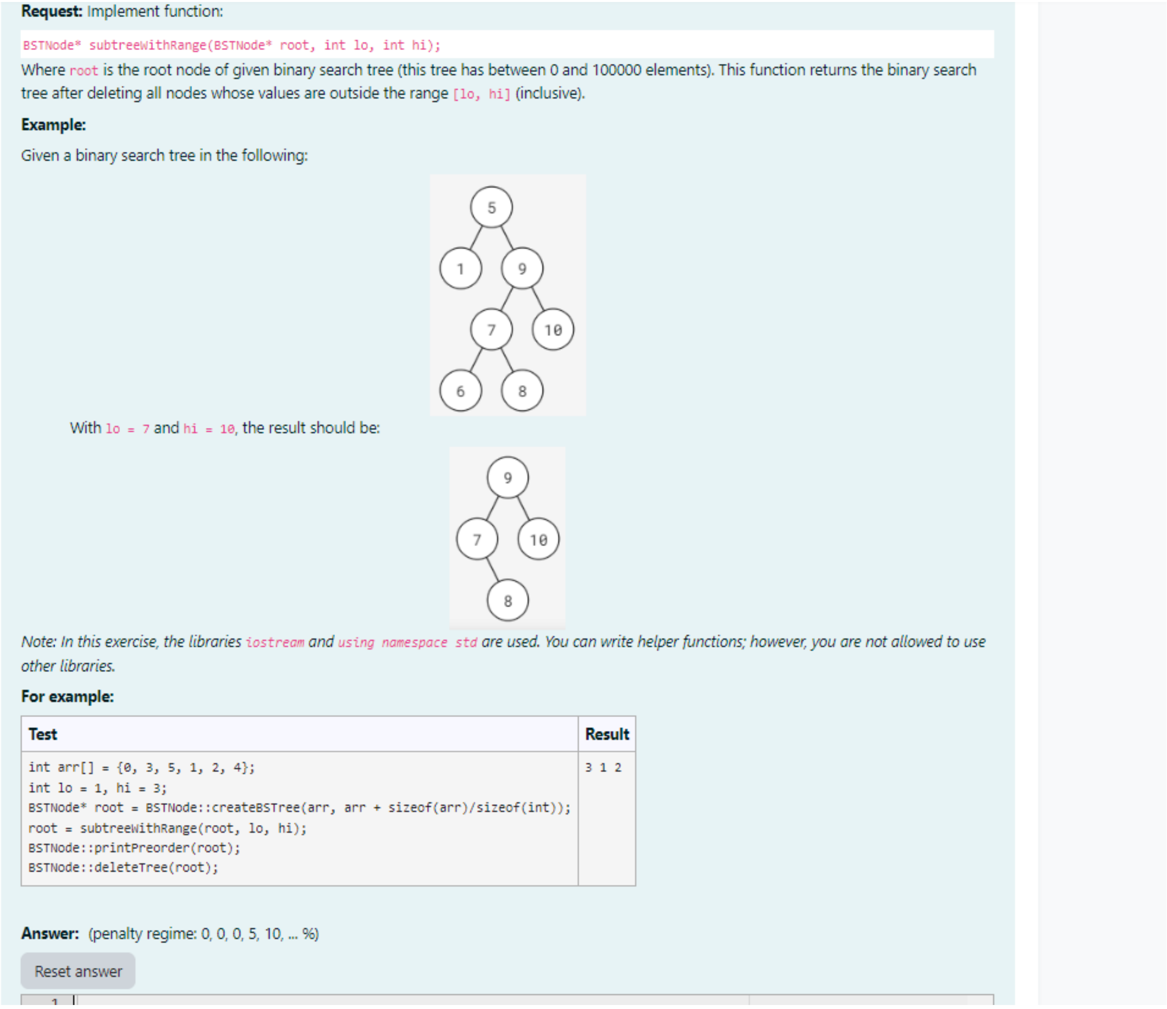
```
BSTNode* subtreeWithRange(BSTNode* root, int lo, int hi);
```

Where root is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the binary search tree after deleting all nodes whose values are outside the range [lo, hi] (inclusive).

Example:

Given a binary search tree in the following:

With lo = 7 and hi = 10, the result should be:



Request: Implement function:

```
BSTNode* subtreeWithRange(BSTNode* root, int lo, int hi);
```

Where root is the root node of given binary search tree (this tree has between 0 and 100000 elements). This function returns the binary search tree after deleting all nodes whose values are outside the range [lo, hi] (inclusive).

Example:

Given a binary search tree in the following:

With lo = 7 and hi = 10, the result should be:

Note: In this exercise, the libraries iostream 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[] = {0, 3, 5, 1, 2, 4};`
`int lo = 1, hi = 3;`
`BSTNode* root = BSTNode::createBSTree(arr, arr + sizeof(arr)/sizeof(int));`
`root = subtreeWithRange(root, lo, hi);`
`BSTNode::printPreorder(root);`
`BSTNode::deleteTree(root);`
	`3 1 2`

Answer: (penalty regime: 0, 0, 0, 5, 10, ... %)