Câu hỏi 1

Chính xác

Chấm điểm của 2,50

In this exercise, you are introduced to 2 inner class **Entry** and **Node** of BTree. You should review your slides to see their structures.

In the template declaration before class BTree, K is a type for key variable, D is a type for data variable, M is the degree of BTree.

Method **toString()** of class Entry is already implemented to print as:

```
<key,data>
```

Your task is to implement method **toString()** of class Node, the return string is of form as:

```
[(n)E1E2...En]
```

- n is the number of entries of node
- E1, E2, ..., En are the string represents each entry respectively, which is the output of **toString()** of **Entry**.

Please refer to example for print format

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

using namespace std;

template <class K, class D, int M> // K: key, D: data, M: degree of BTree
class BTree {
    /// Convention: Left sub-tree < Root's key <= Right sub-tree

public:
    class Entry;
    class Node;

private:
    Node *root;

public:
    BTree() : root(0) {};
    ~BTree() {}

    ///////////////////////////////////////////////////
    ///             CLASS `Entry`                   ///
    ///////////////////////////////////////////////////
public:
    class Entry {
    private:
        K key;
        D data;
        Node *rightPtr;

        friend class BTree<K, D, M>;

    public:
        Entry(K key = K{}, D value = D{}) : key(key), data(value), rightPtr(0) {}
        ~Entry() {}

        string toString() {
            stringstream ss;
            ss << "<"
               << this->key << ","
               << this->data
               << ">";
            return ss.str();
        }

    };

    ///////////////////////////////////////////////////
    ///             CLASS `Node`                    ///
    ///////////////////////////////////////////////////
public:
    class Node {
    private:
        Node *firstPtr;
        int numEntries;
        Entry entries[M - 1];

        friend class BTree<K, D, M>;

    public:
        Node() : firstPtr(0), numEntries(0) {};
        ~Node() { }

        bool isFull() {
            return (numEntries >= M - 1);
        }

        /// BEGIN STUDENT CODE
        string toString() {
            stringstream ss;
            // Fill your code here
            return ss.str();
        }
        /// END STUDENT CODE
    };

    /////////////////////////////////////////////////////////////
    ///         CLASS `BTree`: method run sample test         ///
    /////////////////////////////////////////////////////////////
    void testPrintNode(K* keys, D* data, int size) {
        Node node;

        for (int idx = 0; idx < size; idx++) {
            node.entries[idx].key = keys[idx];
            node.entries[idx].data = data[idx];
        }
        node.numEntries = size;
        cout << node.toString() << endl;
    }
};
```

For example:

Test	Result

`int keys[]  = {3, 5, 7};`
`int data[] = {33, 55, 77};`
`int size = sizeof(keys) / sizeof(int);`
`BTree<int, int, 5>().testPrintNode(keys, data, size);`
	`[(3)<3,33><5,55><7,77>]`

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

Reset answer

```
/// BEGIN STUDENT CODE
string toString() {
    stringstream ss;
    ss << "[" << "(" << numEntries << ")";
    for (int i = 0; i < numEntries; i++) {
        ss << entries[i].toString();
    }
    ss << "]";
    return ss.str();
}
/// END STUDENT CODE
```

Precheck Kiểm tra

	Test	Expected	Got	

✓	`int keys[]  = {3, 5, 7};`
	`int data[] = {33, 55, 77};`
	`int size = sizeof(keys) / sizeof(int);`
	`BTree<int, int, 5>().testPrintNode(keys, data, size);`
		`[(3)<3,33><5,55><7,77>]`	`[(3)<3,33><5,55><7,77>]`	✓

Passed all tests! ✓