ITU Computer Engineering Department
BLG 223E Data Structures, Spring 2022
Recitation #6

Problem Definition

Given two binary trees, which do not contain duplicate nodes and consists of m and n nodes in total, you are asked to find the intersection and union sets of those trees in **O(m+n) expected time**. In this regard, you should be using *unordered_map* which uses hash tables behind. You can view the content of the header files for creating the node structure, generating the binary tree and performing set operations on two binary trees as below.

Code Listing 1: BT Node Header

```
#ifndef BT_NODE
#define BT_NODE

class Node{
    private:
        int data;
        Node* left;
        Node* right;
    public:
        Node(int);
        void set_data(int);
        int get_data();
        void set_left(Node*);
        Node* get_left();
        void set_right(Node*);
        Node* get_right();
};

#endif
```

Code Listing 2: Binary Tree Header

```
#ifndef BT_DEF
#define BT_DEF

#include <vector>
#include "BTNode.h"

using namespace std;

class BT{
    private:
        Node* root;

        void inorder_traverse(Node*);
        void postorder_traverse(Node*);
        void postorder_destruct(Node*);
    public:
        BT();
        ~BT();

        Node* get_root();
        void add(vector<int>);

        void printPostOrder();
        void printInOrder();
};

#endif
```

Code Listing 3: Set Operations Header

```
#include <vector>
#include "BT.h"

#ifndef SET
#define SET

class Set{
    public:
        vector<int> get_intersection(BT*, BT*);
        vector<int> get_union(BT*, BT*);
};

#endif
```

You are only asked to complete the get_intersection and get_union methods of the Set class. The purpose of these methods are provided below:

- vector<int> Set::get_intersection(BT* bt1, BT* bt2)
  - Extracts the values of the nodes that are being shared on both binary trees and stores them in a vector.
  - You need to provide your implementation for this method.
- vector<int> Set::get_union(BT* bt1, BT* bt2)
  - Combines the set of nodes that are present on both binary trees and stores them in a vector.
  - You need to provide your implementation for this method.

You are not allowed to modify the content of the files other than 'Set.cpp' with the purpose of adding new functions. You are free to use the member functions of the BST and Node classes in your code.

Sample Cases

First line of the input is used to fill the first binary tree at the first available position in level order, while the second line is used to fill the second binary tree. Sorting operation is performed in main.cpp to ensure the consistency between the intersecting and union node output and Calico.

Code Listing 4: Test Case 1

```
4 7 9 1 2 34 12 86 3 8 6
6 5 2 46 43 12
```

Code Listing 5: Output 1

```
test@d9d38dbf70fc:~/recitation_6_1_student$ ./bin/main cases/case1.txt
Intersecting nodes
2 6 12
Union nodes
1 2 3 4 5 6 7 8 9 12 34 43 46 86
```

Code Listing 6: Test Case 2

```
5 2 4
8 0 3
```

Code Listing 7: Output 2

```
test@d9d38dbf70fc:~/recitation_6_1_student$ ./bin/main cases/case2.txt
No intersecting nodes found.
Union nodes
0 2 3 4 5 8
```

Submission Rules

- You may want to sync the VSCode configuration files, which you can find them on Ninova. You may also want to change the "args" line in launch.json, of you want to test with a custom input file for debugging.
- Use comments wherever necessary in your code to explain what you did.
- You cannot use stack or queue data structure of STL.
- Your program will be checked by using **Calico(https://bitbucket.org/uyar/calico)** automatic checker.
- Do not share any code or text that can be submitted as a part of an assignment (discussing ideas is okay).
- Only electronic submissions through Ninova will be accepted no later than deadline.
- You may discuss the problems at an abstract level with your classmates, but you should not **share or copy code** from your classmates or from the Internet. You should submit your **own, individual** homework.
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.
- If you have any question about the recitation, you cand send e-mail to Caner Özer (ozerc@itu.edu.tr).
- Note that **YOUR CODES WILL BE CHECKED WITH THE PLAGIARISM TOOLS!**