CS214: Data Structures 2019
Assignment (4) V1.0

-Problems

Section #1 (60 points):

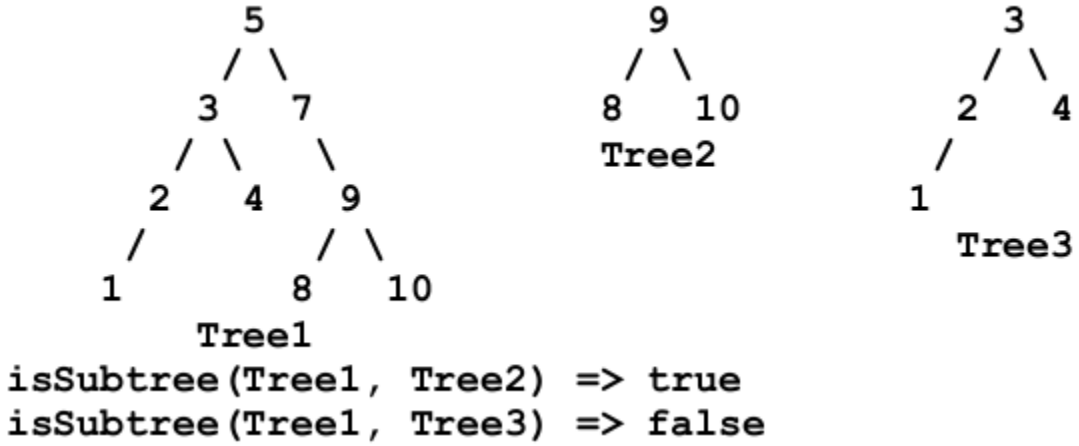
Create template binary search tree class with this name BSTFCI and create node class with name BSTNode (10 points)

1. Add Checking Tree Balance (10 points):

- A Balanced Binary Tree is a tree where the heights of the two child sub-trees of any node differ by at most one and the left subtree is balanaced and the right subtree is balanced.
Add method called **'isBalance'** to BSTFCI this method will check in the BST is balance or not.

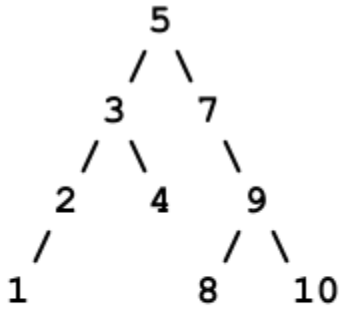
2. Tree Comparison (10 points):

-Write a function that decides if a BSTFCI T2 is a sub-tree of another BSTFCI T1.
Prototype: bool isSubTree(BSTFCI* t1, BSTFCI* t2);
Note: You may need to write another function that takes 2 BSTNodes and compares their sub-trees.



3. Print Range (10 points):

Add a recursive function named **printRange** in the class BSTFCI that stores integers and given a low key value and a high key value, it prints in sorted order all records whose key values fall between the two given keys.
Function **printRange** should visit as few nodes in the BST as possible.
You should NOT traverse all the tree in-order and print the ones in the range. This will not be considered a correct solution. You should do smart traversal to only traverse the related parts.

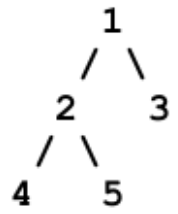



```
printRange(3, 6)    => [3,4,5]
printRange(8, 15)   => [8,9,10]
printRange(6,6)     => []
```

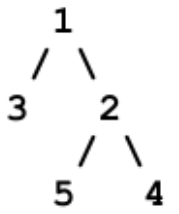
4. Tree Flipping (10 points):

- Write a **flip** method that takes the node which the mirror image of the tree will start from if no parameter send to the function the default value will be the root node.
void flip(Node* node = root)

Example original tree:



Example new tree:



5. Tree Application (10 points):

- Write an index builder application that takes text consisting of lines and prints a list of the words of the text and the lines they appear on are printed next to them.
- The application works by building a binary search tree using BSTFCI and each node contains a word and a vector of that contains the list of lines where this word exists. For each new word, the program finds it and adds the line number to the vector. If word is not found, it is added to the tree. Then traverse the tree in-order to print the nodes.
- You need to remove punctuation marks like **.** and **,** from the text before processing it.
- For example, the text below produces the given index, Test Your code on the given text and 1 more examples.

I am for truth,
no matter who tells it.
I am for justice,
no matter who it is for or against.
Malcolm X

against	4	matter	4
am	1, 3	no	2, 4
for	1, 3, 4	or	4
I	1, 3	tells	2
is	4	truth	1
it	2, 4	who	2,4
justice	3	X	5
Malcolm	5		

Section #2 (40 points):

Contact Manager

Introduction:

- You and your friends are building a Smartphone Operating System (choose its name). You are going to compete with iOS, Android, Blackberry and Windows phone. Competition is very hard, and you need to build strong features. One of the most critical parts of the Smartphone OS is the contact manager, and they decided you are the one to develop it. Are you up to the task?

Problem Statement:

-You are required to implement the contact manager for a smartphone. This is a program that keeps track of all your contacts' information (phone number, email ..) as well as keeping a call log (the recently sent and received calls). You can also quickly reach a contact through search.

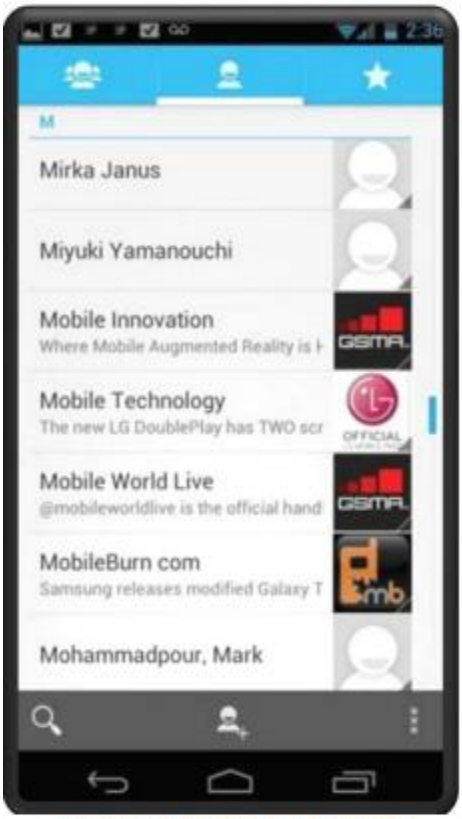


Figure 1: Android ICS Contact Manager

Task 1 (20 points)

-You are given a file with all your contacts. This file is called "allcontacts.in". Each line of the file will be in the following format (without quotes):

"phone number" "contact name"

- Phone numbers contain only digits (**no** + or -) and can start with 0 (e.g. 010...). Contact names are in English (lower-case or upper-case) and **may** contain spaces. A contact name **may** appear more than once in the file, which means this person has multiple phone numbers.

You are required to build a contacts list using an ordered “linked list”, with all the contacts **sorted** by contact name **alphabetically**. Each contact is an object, carrying the name of the contact, and carries all phone numbers as a “linked list”. All phone numbers in the linked list must be **sorted** as well. A phone number will only appear **once** in the file. The linked list should be sorted at all times during the execution (always add elements in sorted order).

After building this data structure, print its contents in the format given below.

Sample "all-contacts.in" file:

002011111 Sayed Hassan

002022222 Mohamed Hesham

002054321 Mina Zaki

002012345 Mina Zaki

002098765 Bob

Sample output:

Bob: 002098765

Mina Zaki: 002012345 - 002054321

Mohamed Hesham: 002022222

Sayed Hassan: 002011111

Task 2 (20 points)

- Once you have prepared the contact list, the call log keeps track of the numbers you recently contacted. Given a list of recently contacted phone numbers, you are required to do this:

1- If the number belongs to a contact, display the contact name.
2- Otherwise display the number unchanged.

The Linked List used in Level 1 is inefficient for this task. We need a better data structure to efficiently search for a phone number and get the name. We will use a “Balanced Binary Search Tree” because it allows us to search in O(log n) while keeping the phone list dynamic. Each node in the tree will carry a phone number and a pointer to the contact. **For this task you are required to implement a "Treap".** It is a Randomized Balanced Binary Search Tree that performs very well in practice. Read the document attached parts 8.1 to 8.4 to understand more about them.

We won't need all of the functionality of the tree for this problem. The required operations are:

- **Add(phone number, pointer to contact):** adds a phone number and pointer to contact in its right position in the tree.

- **Find(phone number):** Searches for a phone number in the tree. If found, returns the contact, otherwise returns null.

Sample input:	**Sample output:**
002098765	Bob
002054321	Mina Zaki
002054321	Mina Zaki
012521212	012521212
002011111	Sayed Hassan