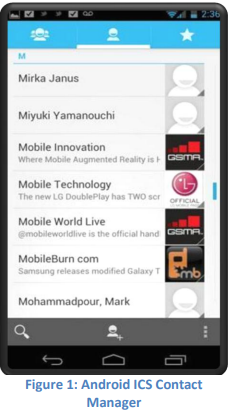
**Contact Manager** 

**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.

**Task 1**

-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**

- 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