Applications

1. Search Application.

**Problem**: Suppose that you are a product manager and currently leading a team to develop a new product. But an error that is unfortunately occurred and lead to your product fails the quality check.

So now, you want to find the first version that made your product’s quality went badly. Since each version is developed based on the previous versions, all the versions after a bad version are also bad.

* You are given an API *bool isBadVersion(version*) to check whether a version is bad or not and you should minimize the number of calls to the API.
* Implement a function to find the first bad version.

**Solution**: You could simply find the bad version by traversing sequencely by checking all the versions until you found the bad one. But this will be time consuming and is not a very effective way.

So, to take the advantage of the property of all the versions after the first bad one are also bad, you can apply the *Binary Search* algorithm to find the first bad version swiftly and reduce the number of times to call the API to check a version.

**Implementation**:



1. Sort Application.

**Application:** A program that works the same with the real-life Book of Phone Numbers. But the user might want to find a person’s phone numbers by their last names or first names alternately. So, the program must allows the user to look up and visualize the list of people with the ascending order of their first or last names.

**Step to implement:**

* Read all the information of a person include: first name, last name, and phone numbers.
* Using any sorting algorithms to sort the list in alphabetical order. (In this demo, I will implement *Insertion Sort*)

**Implementations:**

****

First, let’s define a structure to contain a person’s information.



Build a function to take the data by reading a file.



Finally, Build a sorting function by applying *Insertion Sort* algorithms to sort the list by their first names.



You can also build another function to sort the list by their last names.