Data Structures Midterm (2018-Fall)
Programming Test

You have 100 minutes to solve the problems.
This is an open-note exam. However, you are not allowed to access the Internet other than eTL.
Grading will be done in the Linux environment with Java (OpenJDK) 10, identical to that of the lab.
Do NOT add Korean comment.
If needed, you may add more class variables and/or helper functions.
Make sure your code compiles before submission.
Compress your Java codes (`LinearProbingHashTable.java`, `LinearProbingHashTableTest.java` and `SwitchBox.java`) as `your-studentnumber.zip` or `your-studentnumber.tar` (e.g., 2016-12345.zip).
Write answer for question 2 in the description (at eTL). Upload your zip file to eTL.
Do NOT create subfolder and place your code inside.

Q1. Implement a hash table class `LinearProbingHashTable` (`LinearProbingHashTable.java`) with linear probing supporting the methods `insert`, `remove` and `get` (as discussed in the class). Assume that the keys are integers (but stored in `String`s: to convert the keys to integers, you can use the code discussed in the class.) and the elements are `String`s.

The skeleton code (to implement the table is given partially to implement the table is given partially) is provided in the description file. If needed, you may modify pre-written part of the skeleton code. You are asked to complete it, referring to the instructions given below:

- Use linear probing to handle the overflows (a probe sequence in which the gap between two probes is fixed (usually 1)).
- Define the following methods:
    - `int hashCode(String key)`, to get hash code of a given key.
    - `void insert(String key, String val)`, to insert key-value pair.
    - `String get(String key)`, to get value for a given key.
    - `void remove(String key)`, to remove key and its value.

In addition, write a Java code `LinearProbingHashTableTest.java` that contains a single method `main`, so that it

- reads the input elements from the file `hash.txt`, which has an integer per line. For example:

    [Example `hash.txt`]

    1

    2

    3

    4

- shows the result of hashing, using the pre-implemented function `printHashTable`.

Q2. Test your code with the following hash function:

`h(x) = x mod 1000003`, using a hash table of size 1000003.

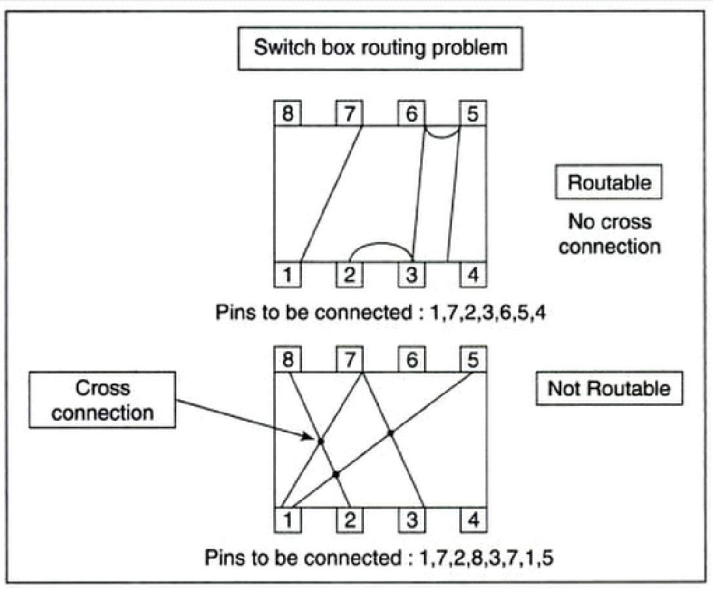
Measure the running time (in nanoseconds) of the `get`, `insert` and `remove` methods for different loading densities: 0.1, 0.25, 0.5, 0.75 and 0.9.

In the description part existing in the submission part of eTL, state the result. For example:

Density 0.1: get 3ns, insert 4ns, remove 4ns
Density 0.25: get 4ns, insert 5ns, remove 6ns
...

Q3. In this problem, you need to write a program `SwitchBox.java` to read a set of integer pairs from a file named `input.txt`, and need to check the validity of the pairs with respect to the SwitchBox Routing problem discussed in the class.

The input file contains a set of integer pairs of the form `x,y`, one on each line. The output (in the screen) should be "`Routable`" if all the pairs can be connected with no `crossing'; otherwise, the output should be two pairs that crossover, each one printed as `(x,y)` separated by a newline character.



The output for the first example in the above figure should be:

[Example 1]

```
cat input.txt
1,7
2,3
3,6
6,5
5,4
java SwitchBox
Routable
```

[Example 2]

```
cat input.txt
1,7
2,8
3,7
1,5
java SwitchBox
(8,2)
(7,1)
```