Csc241 Program 2
Spring 2018

Some points about writing this program

1. **You should be the sole author of this program**. You must draw a line when helping or getting help from another student. **Do not** give other students your code. **Do not** ask other students for their code. **Do not** use code of students who have taken this course previously.
2. Your name should be within the documentation of all source files.
3. Organize your code well.
4. When the assignment is returned I will present my solution, but will not make it available to the class as source code.

This programming assignment involves adding functionality to the `SearchTreeMap` implementation found in the `SetMap` project. Specifically you will edit the file:

```
util.SearchTreeMap
```

and write a Java-correct implementation of the **remove** method.

When you're done, submit the one file, `SearchTreeMap.java`, via D2L. **Please double check that it's the version you want me to grade.**

The goal is to make the `remove` method **behave exactly** like it would for a Java `TreeMap`.

Use the code of the `remove` function in `SearchTreeSet.java` as a conceptual basis, and modify it to work for the map. Avoid stupid approaches like reading the entire tree. In particular, the operation must be O(log(n)) time on the average.

Skeleton Program

Add the following starter code **at the end of the class**, and set **YOUR NAME**:

util.SearchTreeMap

```
//...
public class SearchTreeMap<K, V> extends NavMapAdapter<K, V> {
  // ...
  // Please add the following starter code AT THE END of this class.

  //------------------------ added by YOUR NAME (please set this)
  @Override
  public V remove(Object obj) {
    return null;
  }

}
```

Write a Comparison/Test Program

Create a new package in `SetMap`, like `prog2`, and create a main class to compare the behavior of the Java remove function and your own.

```
package prog2;

import java.util.AbstractMap;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import java.util.Random;
import java.util.TreeMap;
import util.SearchTreeMap;

public class TestRemove {
  public static void main(String[] args) {
    TreeMap<String, Integer> java_map = new TreeMap<>();
    SearchTreeMap<String, Integer> my_map = new SearchTreeMap<>();

    // data to draw from to create random map entries
    String[] names = {
      "jerry", "paul", "erin", "kevin", "helen",
      "bill", "john", "bob", "rick", "tim",
    };

    // create random entries
    List<Map.Entry<String, Integer>> random_entries = new ArrayList<>();
    Random rand = new Random();
    int num_entries = 10;
    String first_key = null;
    for (int i = 0; i < num_entries; i++) {
      String name = names[rand.nextInt(names.length)];   // random name
      if (i == 0) first_key = name;
      int value = rand.nextInt(10);
      Integer age = value == 0 ? null : 10 + value;   // random value or null
      random_entries.add(new AbstractMap.SimpleEntry<>(name, age));
    }

    // create fixed entries
    List<Map.Entry<String, Integer>> fixed_entries = new ArrayList<>();
    fixed_entries.add(new AbstractMap.SimpleEntry<>("john", 22));
    fixed_entries.add(new AbstractMap.SimpleEntry<>("helen", null));
    fixed_entries.add(new AbstractMap.SimpleEntry<>("tim", 17));
    fixed_entries.add(new AbstractMap.SimpleEntry<>("rick", 29));

    List<Map.Entry<String, Integer>> entries;

    //--- choose fixed or random
    entries = fixed_entries;
    entries = random_entries;

    // create the maps from entries chosen
    entries.forEach((entry) -> {
      java_map.put(entry.getKey(), entry.getValue());
      my_map.put(entry.getKey(), entry.getValue());
    });
    String toRemove;

    Integer value;

    //---- choose a name to remove
    toRemove = "john";       // fixed choice
    toRemove = first_key;    // first name of random entries
    toRemove = names[rand.nextInt(names.length)];  // random name

    System.out.println("\n================= java: ");
    System.out.println("before: " + java_map);
    System.out.println("size = " + java_map.size());
    System.out.println("remove: " + toRemove);

    value = java_map.remove(toRemove);

    System.out.println("\nafter: "       + java_map);
    System.out.println("removed value = " + value);
    System.out.println("size = " + java_map.size());

    System.out.println("\n\n================= me: ");
    System.out.println("before: " + java_map);
    System.out.println("size = " + my_map.size());
    System.out.println("remove: " + toRemove);

    System.out.println("----------------");
    my_map.reverseInorderOutput();
    System.out.println("----------------");

    value = my_map.remove(toRemove);

    System.out.println("\nafter: "       + my_map);
    System.out.println("removed value = " + value);
    System.out.println("size = " + my_map.size());

    System.out.println("----------------");
    my_map.reverseInorderOutput();
    System.out.println("----------------");
  }
}
```