Data Structures and Algorithms
INFO 6205
Homework 4
Hongxi Li 001893090
Due: February 10, 2019

Put all your java, compiled class files and documentation files into a zip file named Homework4.zip and submit it via the Drop Box on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. Java String hashcode is the following:
2.

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i}$$

What is the hashCode 32-bit integer number for string =“Hello Students”?

A) Mathematically by hand,

i=0 -> h = 31 * 0 + val[0]
i=1 -> h = 31 * (31 * 0 + val[0]) + val[1]
i=2 -> h = 31 * (31 * (31 * 0 + val[0]) + val[1]) + val[2]
i=3 -> h = 31*(31 * (31 * (31 * 0 + val[0]) + val[1]) + val[2])+val[3]
i=4 -> h = 31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) + val[2])+val[3])+val[4]
i=5 -> h = 31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) + val[2])+val[3])+val[4]
)+val[5]
i=6 -> h = 31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) +
val[2])+val[3])+val[4]
)+val[5])+val[6]
i=7 -> h = 31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) +
val[2])+val[3])+val[4])+val[5])+val[6])+val[7]
i=8 -> h = 31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) +
val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8]
i=9 -> h = 31*(31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) +
val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8])+val[9]
i=10 -> h = 31*(31*(31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) + val[1]) +
val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8])+val[9])+val[10]
i=11 -> h = 31*(31*(31*(31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) +
val[1]) +
val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8])+val[9])+val[10])+val[11]
i=12 -> h = 31*(31*(31*(31*(31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0]) +
val[1])+val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8])+val[9])+val[10])+val[11
])+val[12]
i=13 -> h = 31*(31*(31*(31*(31*(31*(31*(31*(31*(31*(31*(31 * (31 * (31 * 0 + val[0])
+val[1])+val[2])+val[3])+val[4])+val[5])+val[6])+val[7])+val[8])+val[9])+val[10])+val[1
1])+val[12])+val[13]

h= -1752069786

B) Write Java code

```
public int hashCode() {
        int h = hash;
        if (h == 0 && value.length > 0) {
            char val[] = value;
            for (int i = 0; i < value.length; i++) {
                h = 31 * h + val[i];
            }
            hash = h;
        }
        return h;
    }
```

2. Consider the following code for User class.
   A) Discuss code in details
   B) Write Java code to test User class with multiple test cases to test equals, hashCode and CompareTo methods.

```
// A class "User" that implements Comparable
public class User implements Comparable<User> {
        private String name;
        private int id;
        private Date birth;

        public User (String name, int id, Date birth)
                { this.name = name; this.id = id; this.birth = birth; }

        @Override  //override equals function
        public boolean equals(Object other) {
                //whether they are the same object
                if (this == other) return true;
                //Judge the object is null or not, and whether they are the same class
                if (other == null || (this.getClass() != other.getClass()))
                        { return false; }

                User guest = (User) other;
                        return(this.id == guest.id) &&
                                (this.name = null && name.equals(guest.name)) &&
                                (this.dob != null && dob.equals(guest.birth));
        }

        @Override     //override Hashcode function
        public int hashCode() {
                //get the hashcode of the id, name, birth and multiple 31
                int result = 0;
                result = 31*result + id;
                result = 31*result + (name !=null ? name.hashCode() : 0);
                result = 31*result + (birth !=null ? dob.hashCode() : 0);
                return result;
        }

        @Override // Used to sort user by id
        public int compareTo(User o) {
                return this.id - o.id; }
        }
}
```

3. Consider the following example discussed in class for QuickSort, Complete the example
http://interactivepython.org/courselib/static/pythonds/SortSearch/TheQuickSort.html

DBC

4. Consider megeSort algorithm for array {38, 10, 43, 3, 9, 82, 27}. Show the stack operations push and pop step by step for call mergeSort(arr, l, m) and call mergeSort(arr, m+1, r).
Note: I don't need the entire program, just show step by step stack push and pop operations.

If r > l

1. Find the middle point to divide the array into two halves:
   middle m = (l+r)/2
2. Call mergeSort for first half:
   Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
   Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
   Call merge(arr, l, m, r)

Compare	Action	Stack ->Top
10 < 38	Pop 38; Push 10, 38	10, 38
43 > 38	Push 43	10, 38, 43
3<43	Pop 43	10, 38
3<38	Pop 38	10
3<10	Pop 10; Push 3, 10, 38, 43	3, 10, 38, 43
9<43	Pop 43	3, 10, 38
9<38	Pop 38	3, 10
9<10	Pop 10	3,
9>3	Push 9, 10, 38, 43	3, 9, 10, 38, 43
82>43	Push 82	3, 9, 10, 38, 43, 82
27<82	Pop 82	3, 9, 10, 38, 43
27<43	Pop43	3, 9, 10, 38
27<38	Pop38	3, 9, 10
27>10	Push 27, 38, 43, 82	3, 9, 10, 27, 38, 43, 82

5. Consider attached image Boston.jpg. Write a program to sort the image Pixels by "brightness". You program for four sorting algorithms: InsertionSort, HealSort, QuickSort, TimSort, and Merge-Sort. You need to sort the Pixel array size of the image in descending order and show the runtime time complexity of each Sorting algorithm and compare.

Notes:
You may NOT use any Java library function for sorting. You should use ONLY the Sorting Java code I provided in class. The Pixel sorting should start from (0,0) to (high,high) for Brightness. For each Pixel, you need to convert RGB color to appropriate intensity. Use intensity formula: $I = 0.2989R + 0.5870G + 0.1140B$. If the current pixel Intensity is larger than the next pixel intensity, you need to swap, going in descending order.

You may need the following classes:

Java.awt.image.BufferedImage: image class.
eg: image = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
java.util.*: collection of List data types.
javax.imageio.ImageIO: for reading/writing images to file

InsertionSort
Time complexity: O(n^2)
Space complexity: O(n)

HealSort
Time complexity: O(nlog2 n)
Space complexity: O(1)

QuickSort
Time complexity: O(nlog(n))
Space complexity: O(nlog(n))

TimSort
Time complexity: O(nlog(n))
Space complexity:O(n)

MergeSort
Time complexity: O(nlog(n))
Space complexity: O(1)