Project 2: A Few other Sorting Algorithms

The purpose of this project is to understand the syntax of pseudo code and further warm up your Java programming skills.

1 Something to work with...

The following is a complete Java program to test out the insertion sort algorithm.

1.1 A Java implementation of the insertion sort algorithm

Below is an interface, Sorter, and an implementation, InsertionSort, included in the Java Package that comes with the text book by Cormen et al [1], with most of the comments taken off.

```java
public interface Sorter{
    public void sort(Comparable[] array);
}

public class InsertionSort implements Sorter {
    public void sort(Comparable[] array){
        int n = array.length;

        for (int j = 1; j < n; j++) {
            Comparable k = array[j];
            // Insert array[j] into the sorted sequence array[0..j-1].
            int i = j-1;

            while (i >= 0 && array[i].compareTo(k) > 0) {
                array[i+1] = array[i];
                i--;
            }
            array[i+1] = k;
        }
    }
}
```
1.2 A general definition of a comparable type

Below is the Contact.java class, which implements the Comparable type.

//********************************************************************
// Contact.java Author: Lewis/Loftus
//
// Represents a phone contact.
//********************************************************************

class Contact implements Comparable
{
    private String firstName, lastName, phone;

    // Sets up this contact with the specified information.
    public Contact (String first, String last, String telephone)
    {
        firstName = first;
        lastName = last;
        phone = telephone;
    }

    // Returns a description of this contact as a string.
    public String toString ()
    {
        return lastName + ", " + firstName + \t + phone;
    }

    // Uses both last and first names to determine lexical ordering.
    public int compareTo (Object other)
    {
        int result;

        if (lastName.equals(((Contact)other).lastName))
            result = firstName.compareTo(((Contact)other).firstName);
        else
            result = lastName.compareTo(((Contact)other).lastName);
    }
}
1.3 A driver

The following driver should work for the above two classes.

```java
public class SortPhoneList {
    public static void main (String[] args) {
        Contact[] friends = new Contact[7];
        friends[0] = new Contact("John", "Smith", "610-555-7384");
        friends[2] = new Contact("Mark", "Riley", "733-555-2969");

        InsertionSort sorter = new InsertionSort();
        sorter.sort(friends);

        for (int index = 0; index < friends.length; index++)
            System.out.println(friends[index]);
    }
}
```
2 Assignments

1. Study and compare the code of the InsertionSort with the algorithm as presented in pp. 18 of the textbook.

2. Use the given driver to test the above code.

3. Write and test another class, SelectionSort algorithm implementing the selection sort algorithm, as discussed in Exercise 2.2-2. We also discussed it in class. Play with it with some sample data yourself.

4. Implement the BubbleSort algorithm, as discussed in pp. 40 (Problem 2.2). You might want to check the video demonstration of this algorithm, as shown on the course page, and also play with it with some sample data yourself.

5. Implement also the MergeSort algorithm, as discussed in the book. You might want to check the video demonstration of this algorithm, and also play with it with some sample data yourself.

6. Test out all of the works that you have done.

3 What are to be handed in?

Email me the whole java program, including the InsertionSort, SelectionSort, BubbleSort, and the MergeSort classes, and a driver. Also send in a lab report, showing your testing results of the program.

References