Activity 8

Ivan Lizcano

Exercises: 1,7,9,12,13

1. Develop an implementation of Questions.java that takes the maximum number `N` as command-line input. Prove that your implementation is correct.

7. Modify BinarySearch.java so that if the search key is in the array, it returns the smallest index `i` for which `a[i]` is equal to key, and otherwise, it returns `-i`, where `i` is the smallest index such that `a[i]` is greater than key.

9. Write a program Dedup.java that reads strings from standard input and prints them on standard output with all duplicates removed (in sorted order).

12. Add code to LRS.java to make it print indices in the original string where the longest repeated substring occurs.

13. Find a pathological input for which LRS.java runs in quadratic time (or worse).

Creative exercises: 18,19,20,33,34

18. **Median.** Add to StdStats.java a method `median()` that computes in linearithmic time the median of a sequence of *N* integers.

Hint: reduce to sorting.

19. **Mode.** Add to StdStats.java a method `mode()` that computes in linearithmic time the mode (value that occurs most frequently) of a sequence of *N*integers.

Hint: reduce to sorting.

20. **Integer sort.** Write a *linear*-time filter IntegerSort.java that takes from from standard input a sequence of integers that are between 0 and 99 and prints the same integers in sorted order on standard output. For example, presented with the input sequence

```
98 2 3 1 0 0 0 3 98 98 2 2 2 0 0 0 2
```

your program should print the output sequence

```
0  0 0 0 0 0 1 2 2 2 2 2 3 3 98 98 98
```

33. Quicksort. Write a recursive program QuickSort.java that sorts an array of randomly ordered distinct `Comparable` elements.

> *Hint:* Use a method like the one described in the previous exercise. First, partition the array into a left part with all elements less than *v*, followed by *v*, followed by a right part with all elements greater than *v*. Then, recursively sort the two parts.
>
> *Extra credit:* Modify your method (if necessary) to work properly when the elements are not necessarily distinct.

34.Reverse domain. Write a program to read in a list of domain names from standard input, and print the reverse domain names in sorted order. For example, the reverse domain of `cs.princeton.edu` is `edu.princeton.cs`. This computation is useful for web log analysis. To do so, create a data type Domain.java that implements the `Comparable` interface, using reverse domain name order.