Section Handout 5

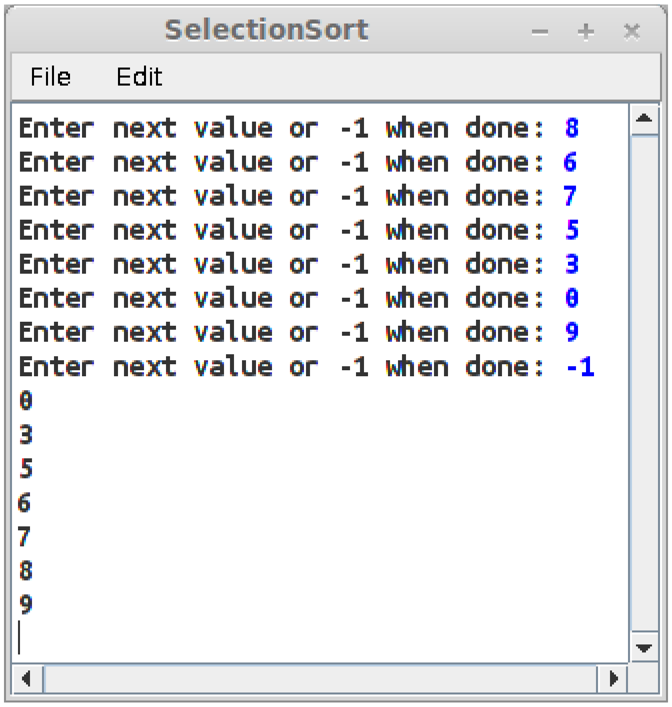
Problem One: Selection Sort

Write a program that reads a list of integers from the user, then prints them out in sorted order. The user should be able to enter values until they enter a sentinel value (say, -1), at which point the program prints the numbers back in sorted order.

To sort the numbers, you should use the *selection sort* algorithm, which works as follows:

- Find the smallest number in the list.
- Remove that number from the list and display it.
- Repeat until no numbers remain.

Here is a sample run of the program:



```
Enter next value or -1 when done: 8
Enter next value or -1 when done: 6
Enter next value or -1 when done: 7
Enter next value or -1 when done: 5
Enter next value or -1 when done: 3
Enter next value or -1 when done: 0
Enter next value or -1 when done: 9
Enter next value or -1 when done: -1
0
3
5
6
7
8
9
```

Problem Two: Method Testing

Below are some descriptions of methods. Your job is to describe how you would test those methods to make sure that they work correctly. What inputs would you feed into them? Why would those inputs form good test cases?

```
/**
 * Given a string representing a single word, estimates the number of syllables
 * in that word by counting the number of groups of vowels, except for isolated
 * e's at the end of the word.
 *
 * @param word The word in question
 * @return An estimate of the number of syllables as described above.
 */
private int syllablesInWord(String word)

/**
 * Given two numeric strings (strings consisting purely of digits) representing two
 * numbers n1 and n2, returns a numeric string representing their sum. The input
 * strings don't have to be the same length, but each will represent a nonnegative
 * integer.
 *
 * @param n1 The string encoding of the first number
 * @param n2 The string encoding of the second number.
 * @return A string encoding of their sum.
 */
private String addNumericStrings(String n1, String n2)

/**
 * Given as input a string of three letters and a list of strings, returns all
 * words in English that match that word according to the rules of the “license
 * plate game” (that is, all words that contain all of the letters in the string
 * 'letters' in the order in which they appear).
 *
 * @param letters A string of three letters.
 * @param allWords A list of all the strings to test.
 * @return A list of all the English words matching the given letter patter.
 */
private ArrayList<String> allMatchesFor(String letters, ArrayList<String> allWords)

/**
 * Tokenizes the input CSV file line and returns all the fields in that line.
 *
 * @param line A line from a CSV file.
 * @return A list of all the tokens in that line, with any external quotation marks
 *         removed.
 */
private ArrayList<String> fieldsIn(String line)
```