ListMagic.java ListMagic.txt

In a program named ListMagic, write the following methods.

Write a method `maxLength` that takes an `ArrayList` of `String`s as a parameter and that returns the length of the longest string in the list. If your method is passed an empty list, it should return 0.

Write a method `removeDuplicates` that takes as a parameter an `ArrayList` of `String`s and that eliminates any duplicates from the list. For example, suppose that a variable called `list` contains the following values: `{"be", "be", "is", "not", "or", "question", "that", "the", "to", "to"}` After calling `removeDuplicates(list);` the list should store the following values: `{"be", "is", "not", "or", "question", "that", "the", "to"}`

Because the values will be sorted, all of the duplicates will be grouped together.

Write a method `range` that accepts an `ArrayList` of integers as a parameter and that returns the range of values contained in the list, which is defined as 1 more than the difference between the largest and smallest elements. For example if a variable called `list` stores the following values:

```
[18, 14, 29, 12, 7, 25]
```

The call of `range(list)` should return `23`, because this is one more than the largest difference between any pair of values (29 - 7 + 1 = 23). An empty list is defined to have a range of 0.

Write a method `switchPairs` that switches the order of values in an `ArrayList` of `String`s in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, if the list initially stores these values: `{"four", "score", "and", "seven", "years", "ago"}` your method should switch the first pair, `"four", "score"`, the second pair, `"and", "seven"`, and the third pair, `"years", "ago"`, to yield this list: `{"score", "four", "seven", "and", "ago", "years"}`

If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been: `{"to", "be", "or", "not", "to", "be", "hamlet"}` It would again switch pairs of values, but the final value, `"hamlet"` would not be moved, yielding this list: `{"be", "to", "not", "or", "be", "to", "hamlet"}`

Write a method called `removeInRange` that accepts four parameters: an `ArrayList` of integers, an element value, a starting index, and an ending index. The method's behavior is to remove all occurrences of the given element that appear in the list between the starting index (inclusive)

and the ending index (exclusive). Other values and occurrences of the given value that appear outside the given index range are not affected.

For example, for the list `[0, 0, 2, 0, 4, 0, 6, 0, 8, 0, 10, 0, 12, 0, 14, 0, 16]`, a call of `removeInRange(list, 0, 5, 13);` should produce the list `[0, 0, 2, 0, 4, 6, 8, 10, 12, 0, 14, 0, 16]`. Notice that the zeros located at indices between 5 inclusive and 13 exclusive in the original list (before any modifications were made) have been removed.

In the main method, Create an ArrayList of Strings and prompt the user to fill until the word "Quit" is entered. Create a second ArrayList of Integers and fill with Integers until a negative value is input.

Print the list of Strings

Invoke maxLength, printing the returned value.

Invoke removeDuplicates, printing the resulting list.

Invoke switchPairs, printing the resulting list.

Print the list of Integers

Invoke the method range, printing the returned value

Prompt the user for an element value, a starting index, and an ending index and use to invoke the method removeInRange, printing the resulting list.

Sample Output

```
Enter word, enter quit to end
be be is not or question that the to to
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
Enter word, enter quit to end
quit
Enter num, enter negative num to end
18 14 29 12 7 25 -5
Enter num, enter negative num to end
Enter num, enter negative num to end
Enter num, enter negative num to end
Enter num, enter negative num to end
Enter num, enter negative num to end
Enter num, enter negative num to end
[be, be, is, not, or, question, that, the, to, to]
8
[be, is, not, or, question, that, the, to]
```

```
[is, be, or, not, that, question, to, the]
[18, 14, 29, 12, 7, 25]
23
Enter element, starting index, and ending index
12
3
5
[18, 14, 29, 7, 25]
```