

# Review Exercises

---

Here is a list of review exercises. There are times when an exercise will be open ended, allowing you to pursue multiple aspects of the material. While encompassing some areas we have studied, you should further supplement your studying by reviewing: your notes, your project assignments, exercise questions given in lecture, exercise questions given in discussion sessions, and problems from the chapters in our textbook.

1. Create the *analysis*, *design*, and *implementation* of a method *diamond*. The method will print a diamond figure to the screen. The method will receive an int for the *length* of the diagonals to print. The method will not return anything. If five were passed to the method, the output would be:

```
  1
 2  2
3   3
 4  4
  5
```

2. Create the *analysis*, *design*, and *implementation* of a method *diamond*. The method will print a series of diamond figures to the screen. The method will receive an int for the *length* of the diagonals to print and an int for the *number* of diamonds to be printed. Similar to Question #1, if the user entered five and the three, the output is shown here. The method will not return anything.

```
  1      1      1
 2  2    2  2    2  2
3   3  3  3   3  3   3
 4  4  4  4  4  4  4
  5      5      5
```

3. Create the array *sentence*, type char, and ask the user for the number of characters to be entered. Then receive these characters, placing them within *sentence*.
4. Create the *analysis*, *design*, and *implementation* of the method *copy*. The method will copy the contents of an array into a new array. The method will receive the array to be copied, the size of the new array, and return the newly created array.
5. Create the *analysis*, *design*, and *implementation* of the method *shrink*. The method will format the size of the array *sentence* to only be the size of the number of spaces actually used within the array. Suppose you modify your input questions to the user, allowing another character to be entered or the stopping of character input. Recall a position in a char array that has not be initialized is equal to "", that is two single quote with nothing in between. A *null* or empty character, opposed to a blank space.

6. Create the *analysis*, *design*, and *implementation* of the method *duplicates*. The method will receive a String and return a HashMap. The *keys* in the HashMap will be Strings. The *values* in the HashMap will be Integers. The HashMap will contain each substring and a count for the number of times the substring is found in the String given to the method. For example, given “two one two four one three one four one”, the HashMap will contain (“two”, 2), (“one”, 4), and so on.
7. Create the *analysis*, *design*, and *implementation* of the method *commaSeparate*. The method will receive an int value and modify the numeric representation to include comma’s. Use the standard mathematical definition for comma separating a numeric value, i.e. every third digit is separated from the other digits by a comma. If 7654321 were given to the method, “7,654,321” would be returned.
8. Create the *analysis*, *design*, and *implementation* of the method *printJustifiedList*. The method will receive an ArrayList containing Integers. Each Integer will be printed to the screen justified to the right of the page. If the ArrayList contained: 14, -238, 3, 123456, and 1661 the output would be:

```
    14
  -238
     3
123456
1661
```