Text manipulation is very very common in programming. Luckily, Java has a lot of built-in library functions that can do a lot of the work for you.

If s is a String, the most common methods built into s are:

s.length() - returns the number of characters in s.
s.charAt(i) - returns the character at index i of s (indices begin at 0).
s.substring(i,j) - returns the substring of s with indices from i to j-1.
s.substring(i) - returns the substring of s with indices from i to the end of s.
s.equals(t) - returns true if Strings s and t are the same. (Don't use ==)

There are many more. Find the Javadoc for the String class and have a quick look.

1) Let's say s is a String. Write a few lines of code that print the characters of s, one per line, using charAt.

2) Let's say s is a String. Write a few lines of code that print the characters of s, one per line, using substring.

3) Write a small program that prompts the user for a word and then prints all the ways you can break the word into two non-empty strings with a "|" character separating the two parts. For example:

   ```
   Enter a word: abcd
   a|bcd
   ab|cd
   abc|d
   ```

   Before coding, design your algorithm first using pseudocode. Using paper and pencil and examples can be the best way to figure out important details such as the indices to use for your substrings. Ask yourself questions like "how many times should I loop?" and "what sequence of loop indexes i do I want?".

A String s is a rotation of another String t if you can move some characters from the front of t to the back of t and the result matches s. For example, cdab is a rotation of abcd because if you remove the ab from abcd and move it to the back you get cdab.

For each of the following problems you can allow or disallow calling a String a rotation of itself. Some of them are easier if you do count it.

4) Write a small program that prompts the user for two words and then prints either "Rotation!" or "No rotation!" depending on whether they are rotations of each other. Before coding, design your algorithm first using pseudocode.

5) How can you convince yourself that the program you just wrote works correctly? Testing.

   In programming, having a test plan is very important. You should consider a selection of inputs that convince you that it works well. Try normal inputs, exotic inputs, tiny inputs, etc. Both ones that should succeed and ones that should fail. You don't need dozens of test cases, but six or eight well-chosen ones should suffice.

   Discuss with others what would make a good set of test inputs for your program.

6) Another (more efficient) way to check whether s is a rotation of t is to check whether s and t are the same length AND s is a substring of t+t. For example we see that cdab is a rotation of abcd because both are length 4 and cdab is a substring of abcdabcd. Redo Problem 4 using this logic and looping through all the correct-length substrings of t+t, if s and t are the same length. Before coding, design your algorithm first using pseudocode.

7) Another (even more efficient) way to check whether s is a rotation of t is to follow the logic of Problem 6, but having Java do the looping for you. Read about the indexOf method in the String class Javadoc and use it instead of the looping you did in your Problem 6 solution. Before coding, design your algorithm first using pseudocode.