Solution to Practice Problems: String Problems

0. All Primitive Data Types
Answer the following questions about the primitive data types:

a. What types of values can legally be assigned to a short variable without a type conversion? What about to a long variable? A double variable?

shorts can be assigned only byte and short values. longs can be assigned all discrete types except booleans: longs, ints, shorts, chars, and bytes. doubles can be assigned all primitive types except booleans: floats, longs, ints, shorts, chars, and bytes.

b. How many bits are needed to store each primitive data type? How many bytes?

booleans: 1 bit. bytes: 8 bits, 1 byte. chars: 16 bits, 2 bytes. shorts: 16 bits, 2 bytes. ints: 32 bits, 4 bytes. longs: 64 bits, 8 bytes. floats: 32 bits, 4 bytes. doubles: 64 bits, 8 bytes.

c. How many different values can be stored in each of the primitive data types?

2 to the power of number of bits for the data type

d. What is the value of this expression: (char) ('r' + 1)

's'

e. Name as many differences between a String variable and a char variable as you can.

1) Strings are references, chars are not.
2) Strings are object types, chars are primitive types.
3) Strings can hold between 0 and any number of characters. char variables hold exactly 1 character.
4) Adding an int and a char creates an int value (first typecasting the char to an int). Adding an int and a String does String concatenation.
5) Strings have methods, like length() and indexOf(). chars (like all primitive types) have no methods.
6) There's probably more, but I can't think of any right now.

1. Tracing Code with Strings
Show what is stored in memory at the end of each of these programs.

```java
public class String.Assignments {
    public static void main(String[] args) {
        String s;
        String t = null;
        String u = "you";
        String v = new String("me");
        String w = u + v;
    }
}
```

```plaintext
s t u v w
null
"you"
"me"
"youme"
```
```java
class StringCommands {
    public static void main(String[] args) {
        String s = "Call me Ishmael.";
        int len = s.length();
        int ishPos = s.indexOf("Ish");
        int jackPos = s.indexOf("Jack");
        String ishSub = s.substring(ishPos, len);
        char c = s.charAt(ishPos);
    }
}
```

// Here is an example that removes a portion of a String,
// and inserts a replacement
```java
class StringInsertDelete {
    public static void main(String[] args) {
        String s = "It was a bright cold day in April, " +
                "and the clocks were striking thirteen.";
        int startThirteen = s.indexOf("thirteen");
        int endThirteen = startThirteen + "thirteen".length();
        s = s.substring(0, startThirteen) + "twenty-five"
           + s.substring(endThirteen, s.length());
    }
}
```

// Here is a typical example of a loop used to
// process a String.
// In this example, the loop visits each character
// in the String once.
```java
```
public class String-Processing {
    public static void main(String[] args) {
        String s = "Call me Ishmael.";
        int aCount = 0;
        for(int i=0; i<s.length(); i++) {
            char c = s.charAt(i);
            if(c == 'a') {
                aCount++;
            }
        }
    }
}

// Here is an example that repeatedly loops through the String, processing one word at a time.
public class String-Processing {
    public static void main(String[] args) {
        String s = "Ships at a distance have every man's wish on board."
        int spacePos1 = 0;
        int spacePos2 = s.indexOf(" ");
        String hyphenated = ";
        while(spacePos2>=0) {
            String word = s.substring(spacePos1, spacePos2);
            hyphenated = hyphenated + word + ";
            spacePos1 = spacePos2 + 1;
            spacePos2 = s.indexOf(" ", spacePos1);
        }
        if(spacePos1<s.length()) {
            hyphenated = hyphenated + s.substring(spacePos1, s.length());
        }
    }
}


2. **Repeat-X and Sum Algorithms with Strings**

Write a short Java program to solve each of the following problems. Each one will involve a String, plus a Repeat-X or an accumulate algorithm (and maybe more than one) --- it's up to you to figure out how!

1. **Read a String from the keyboard, and count how many letter 's' or 'S' are in the String that the user enters.**

   ```java
   import java.util.Scanner;
   public class CountSs {
      public static void main(String[] args) {
         Scanner kb = new Scanner(System.in);
         String str = kb.next();
         int numS = 0;
         for(int i=0; i<str.length(); i++) {
            char c = str.charAt(i);
            if(c == 's' || c == 'S') {
               numS++;
            }
         }
         System.out.println(numS + " s's and S's in your String");
      }
   }
   ``

2. **Read 10 Strings from the keyboard, and compute their total length.**

   ```java
   public class SumLengths {
      public static void main(String[] args) {
         Scanner kb = new Scanner(System.in);
         int totalLength = 0;
         for(int i=0; i<10; i++) {
            String str = kb.next();
            totalLength = totalLength + str.length();
         }
         System.out.println("Your strings have total length = " + totalLength);
      }
   }
   ```