COMP 2150 – Spring 2014
Homework 1: String Manipulation
(Posted Jan. 29 – Due Feb. 5 by 12:40 pm)

Remember that as stated on the syllabus, this assignment should be an individual effort (contact me or visit the Computer Science Learning Center, www.cs.memphis.edu/cslc, if you need help).

Submission: Please zip your source code into a single file (you can zip the entire project folder if you're using BlueJ) and upload it to the proper folder in the eCourseware dropbox at https://elearn.memphis.edu. The dropbox will cut off all submissions after the indicated deadline, so don't wait until the very last minute to submit your work!

Grading: This assignment will be graded by the TA Vincent Nkawu (venkawu@memphis.edu). If you have questions or concerns about your grade, please contact him first. I'll be happy to look over your assignment myself if he's not able to resolve the situation to your satisfaction. Also remember that as stated on the syllabus, all submissions MUST compile and run to receive credit. The TA does not have time to find and correct your syntax errors!

Coding Style: Be sure to follow the good coding practices that were discussed in COMP 1900:

- Use descriptive variable and method names. Avoid using single-character names unless it's very obvious what the variable's being used for (like a loop counter).
- Follow standard Java programming conventions for **variableAndMethodNames**, **ClassNames**, **CONSTANT_NAMES**.
- Consistently indent your code.
- Use comments judiciously. Every source code file should include a comment block at the top that lists your name and the assignment number. Every non-trivial method should have a comment preceding it that specifies what the method does, what its parameters are, and what it returns. For simple methods like accessors or mutators this isn't necessary, but it doesn't hurt to put in a simple comment like `// accessor methods`. I also expect to see comments throughout your code explaining what actions are being taken!

This assignment deals with string manipulation problems.

- Use only the **charAt**, **equals**, **length**, and **substring** methods of the **String** class (see the Java API for details) in your solutions.
- The methods you're writing for this assignment simply make use of **String** objects. You're not actually writing new methods for Java's built-in **String** class! All your methods can be declared as **static**, just like most of the ones you wrote from COMP 1900.

1. **(4 pts)** Write a method **toIntArray(String s)** that returns an array of the ASCII values of the characters in the parameter **s**. (Hint: remember that you can cast a **char** value to **int** to get its ASCII value.)

   Example:

   `toIntArray("hunter2")` should return the array {104, 117, 110, 116, 101, 114, 50} (the ASCII value of 'h' is 104, the ASCII value of 'u' is 117, and so on)

For problems 2-4: Honors students should do all three **(7 pts each)**. Regular students can pick any two **(10.5 pts each)**.

2. You've set up a My Little Pony fan website that collects registration information from visitors via a form. This information is stored as strings in a database, and you want to ensure that it's consistently formatted. Specifically, you want to eliminate any leading and trailing spaces, and eliminate any extra spaces within the body of each string. Note that you're not removing ALL spaces from the string – only the extraneous ones!

   Write a method **trim(String s)** that returns a version of the parameter **s** with all extra spaces removed as described above. Calling **trim** with a string of all spaces as the argument should return the empty string (i.e., the string containing no characters).

   

   Examples:

   ```
   trim("     Rainbow   Dash")
   trim("Rainbow     Dash    ")
   trim(" Rainbow Dash ")
   ```

   should all return the string `"Rainbow Dash"`

   (Yes, I'm aware that's not Rainbow Dash in the picture. I am not particularly proud that I know this.)

3. On your website's registration form from the previous problem, one of the pieces of information is the user's phone number. You plan to use this for all kinds of nefarious purposes. Exciting!

   Write a method **isValidPhoneNumber(String n)** that determines whether or not the string **n** is in valid phone number format. Assume a "valid" phone number is something in the form DDD-DDD-DDDD, DDD.DDD.DDDD, DDD DDD DDDD, or DDDDDDDDDD, where D is a digit from 0-9. These four formats cannot be mixed. (If you happen to be familiar with regular expressions, do not use them for this problem. That would be too easy ☺)

   Examples of *valid* phone numbers:
   901-867-5309
   901 867 5309
   9018675309

   Examples of *invalid* phone numbers:
   901-867.5309
   901867 5309
   +4f 89w efg6

4. Write a method **findLongestPalindrome(String s)** that returns the longest palindrome (string that matches its own reverse) contained within the parameter **s**. If **s** contains more than one palindrome of the same length, any one of them can be returned. (This problem is based on an actual interview question asked by Microsoft!)

   Examples:

Argument	**Return**
`"radaramanaplanacanalpanama"`	`"amanaplanacanalpanama"`
`"bob"`	`"bob"`
`"boba"`	`"bob"`
`"a"`	`"a"`
`"xyz"`	`"x"` (`"y"` or `"z"` would also be acceptable)
`"a radar racecar astonmartin"`	`" racecar "`
`""` (the empty string)	`""`