CISC 181 Lab 2 (100 pts)

Due: March 7 at midnight (This is a two-week lab)

This lab may be done individually or with a partner. Working with a partner DOES NOT mean, "you do the evens, and I'll do the odds". Choosing an irresponsible partner is not an excuse for not finishing the lab on time. Labs are to be turned in via Sakai by midnight on Monday, March 7 (the midnight between Monday and Tuesday). Late labs will receive 10% if they are turned in on Tuesday before midnight. After that, labs will no longer be accepted.

Again, I ENCOURAGE students to help each other. If you get something and you can help another student, please do! If you don't understand something, please ask! However, it doesn't do anyone any good to turn in labs that you don't understand.

Please consult chapters 1-6 of your textbook, and your class notes for additional examples that will help with concepts found in these problems. Note: The review problems at the end of each chapter generally contain enough code to guide you to an answer, which is great practice. Also, the solutions for even numbered problems are available at http://www.cs.armstrong.edu/liang/intro9e/exercisesolution.html.

- Create a class named Lab2a. This class will consist solely of *public static* methods. Everything should be called in your main function (method).

- Please place a comment before each method describing the method and what it does. You do not need to comment individual lines of code in this assignment.

Helpful Java for Lab2 (Part B):

- arr.length will give you the number of elements in the Array arr, regardless of whether the array is of ints, chars, Strings, or doubles.
- You can use Math.pow(x,y) to get x to the power of y, and you can use Math.abs(x) to get the absolute value of x
- You can convert a String to an array of characters using toCharArray(). So, for instance, if you have the String "hello" you can convert it to an array of {'h','e','l','l','o'} by simply writing:

  "hello".toCharArray();

---

1. **[6 pts]** Loops and Multiplication: Multiplication of 2 numbers can be done using a loop and just addition. Write 3 functions. Each function takes as input parameters 2 integers and returns an integer. Each function returns the value of the 2 numbers multiplied together. Each number uses only addition and a loop. The first uses a while loop. The second uses recursion. And the third uses a for loop.
2. **[6 pts]**Calculating the LCM using loops. The Least Common Multiple (LCM) of 2 numbers is the smallest number that two numbers divide into evenly. So, for instance, if the numbers were 22 and 12, the LCM would be 132.

Write a function that takes as input parameters 2 integers and returns an integer. The integer being returned is the lcm of the two input parameters. It should use a while loop to calculate the LCM. (Note: One often calculates the LCM of 2 numbers using the prime factors. You really don't have to work that hard. Calculate this in a way that makes sense to you).
Write a second function that uses a for loop to calculate the LCM.

3. **[9 pts]** In order to calculate the square root of a number, we can always use java's built-in Math.sqrt(x) method. But someone somewhere had to write java's Math.sqrt method. Why not you?

   So you will be writing a function called **getSquare,** that takes as an input parameter an integer, and returns a double. We calculate the square root of a number using the following formula

   $$sqrt_{n+1} = \frac{sqrt_n + \frac{num}{squrt_n}}{2.0}$$

   Where num is the number we're trying to find the square root of. We want to continue doing the above formula until the sqrt is within a certain acceptable range of the actual square root. In other words, if our acceptable range is 0.0001, then we want to keep calculating a new sqrt until sqrt x sqrt is within 0.0001 of the num value. If you want a more precise square root, just increase the precision of the range.

   For our purposes, the very first sqrt number should be the input number divided by 2.0.

   Hints: You'll need to use a loop. And you'll want to use java's built-in absolute value calculater, e.g., Math.abs(x).

4. **(10 pts)** Write a function that will print a square as follows: It takes 6 input parameters: 2 integers and 4 characters. It then prints the first character in the upper left corner, the second character in the upper right corner, the third character in the lower left corner, and the 4th character in the lower right corner. It prints a space to separate each of these corners. The size of the corners is determined by the first input parameter, and the size of the entire square is determined by the second input parameter. So, for instance, if the input parameters are: 4,10,'a','b','c',and'd', the resulting square should look like this:

```
a a a a    b b b b b
a a a a    b b b b b
a a a a    b b b b b
a a a a    b b b b b

c c c c    d d d d d
c c c c    d d d d d
c c c c    d d d d d
c c c c    d d d d d
c c c c    d d d d d
```

   Whereas input parameters of 8, 11, '*', '!', '+', '@', should look like this:

```
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !
* * * * * * * * *    ! !

+ + + + + + + + +    @ @
+ + + + + + + + +    @ @
```

Hint: for printing, you'll need to use System.out.print(c) to print each character in the row, without going to a new row after each character is printed. You can also use System.out.println() to print a new line.
Note: this function doesn't return anything.

5. **(5 pts)** Given two ints, each in the range 10..99, return true if there is a digit that appears in both numbers, such as the 2 in 12 and 23.

   Write an implementation for a static method, **`shareDigit`**, that implements this logic **without using a conditional statement (no if, no switch, no ternary operator)**.

6. **[10 pts]** Many digital identification numbers such as ISBNs, credit card numbers, bank routing numbers, and UPCs are designed with an extra digit that helps detect (and sometimes correct) human entry errors. In this problem we will assume a simple check-digit algorithm:

   All correct identification numbers should have odd parity. Odd parity is determined by adding all of the digits of the number and checking to see if the result is odd.

   4532120 is odd parity because 4+5+3+2+1+2+0 = 17 which is odd, whereas
   4532121 is not odd parity because 4+5+3+2+1+2+1 = 18 which is even

   Write an implementation for a static method, **`oddParity`**, that implements this logic using a while loop. This method will return a Boolean value.

****

Part 2:

7. **[6 points]** Write the method:

   public static double mean(double[] data)

   where the returned double is the average of all values in data array.

8. **[8 points]** Write the method **countWords**. Given a sentence as an array of characters, compute how many words are in the sentence. The only characters that will appear in our sentences are letters, spaces, commas, and a period will always end the sentence. Every comma will be followed by a space.
   - Make sure you include at least 3 test cases for test_countWords

9. **[10 points]** TVs today are often able to display images at very high definition. A problem, however, is that many tv shows and movies are recorded in a lower definition than the TV displays. In order to create a higher definition image, numbers are added by finding values that fall between two recorded values. For instance, if an image's digital representation is:

   {400, 500, 600}

   And the TV's definition is 3 times the image's definition, we'd create an image signal that would be:

   {400,433,466,500,533,566,600}

   Your job: Write a method that takes as input an array of integers and returns an array of integers 3 times (minus 2) as long as the input array. The returned array should be the high definition array, with values 1/3 and 2/3 between the two values. (Note: I always just rounded down)

   - Make sure you include at least 3 test cases for test_highDef

10. **[12 points]** In my research, I often need to count or find words in a document to help me determine the general meaning of a document, or even how related two words are semantically. In order to do this, I need to strip the punctuation out of a document first so that I know that "computer" and "computer!!" are the same word.

    Your job: create a method called **stripPunct** that takes as input a character array representing a sentence with punctuation. You can assume the only punctuation in the sentence is period, comma, question mark, and exclamation mark. The function should return a character array without any of the punctuation in it.

    - Make sure you include at least 3 test cases for test_stripPunct

    Extra Credit **(10 pts**) Modify the above method so that, instead of returning an array of characters representing the sentence stripped of punctuation, it returns an array of strings, with each string being a word in the sentence. Make sure you add test cases to the Lab2aTest.java file.

11. **[18 points] Simple Compression:** All files (sound, images, text, etc.) are stored as numbers, and even more specifically binary numbers. This can take up huge amounts of space, not to mention taking vast quantities of resources when transferred across the internet. For

example, a typical sound file samples displacements of air 40,000 times per second. The sheer size of a sound file could be overwhelming. Thus we spend a lot of time trying to figure out ways of compressing data.

A ridiculously simple way of making the size of a file smaller is to take an array of numbers (e.g., the measurements of displacement of air) and, instead of storing each of those numbers, storing the difference between the current number and the previous number (measurement) in the array. So, for instance, if you had a sound file that had the following measurements:

4283, 4296, 4344, 4348, 4355, 4367, 4392, 4391, 4380

You would store the following numbers:

4283, 13, 48, 4, 8, 12, 25, -1, -11

The savings in the number of digits stored would be: 19 (we don't need to worry about the -), or 36 digits becomes 17 digits (a compression of over 50%). (And this is a ridiculously simple compression technique that no one really uses anymore. Think how much we could save if we actually did something with some intelligence!!)

Your job: Write a method that takes as input an array of integers and returns a string. The method should create a brand new array (because what we really want is the new, compressed array) that contains the first number in the old array, and then the difference between the current number in the array and the previous number in the array.

It will return a string that will say

"Saved ***x*** in digits"

Where x is the actual number of digits saved (so in the example above the returned string would be:

"Saved 19 in digits"

Make sure you include at least 3 test cases.

A couple of notes: To make this work, I created a second function that calculated the number of digits in a number and returned that value.

Also, while I was interested in how many digits we saved using our compression technique, we'd usually want the actual compressed array. Thus your method must create an array of the differences. This is where classes will come in quite handy – with a class we could return both the compressed array and the number of digits saved.

Finally, this method returns a string.

Turn in to Sakai your entire archived Eclipse Lab2 project as follows:

1. Select project
2. File-> Export -> General -> Archive File
3. Click Browse to find place on your computer to place archive
4. Upload archive to Sakai

You should be uploading a single archive file and then clicking “submit”.