Honor Code

Like all work for this class, the Emory Honor Code applies. You should do your own work on all problems, unless you are explicitly instructed otherwise. If you get stuck or have questions, ask your instructor or a TA for help. For EACH of the files you submit, be sure to put the appropriate honor code statement (as specified on the course syllabus) at the top of the file in comments. Otherwise, you will be assessed a 10 point deduction.

Preparation

1. Create a directory called hw5 inside your cs170 project directory to save your hw5 files.
   
   mkdir ~/cs170/hw5

2. You must use /cs170/hw5 directory as your current directory when editing any program files for hw5. Copy the files from the shared directory:

   cp /home/cs170002/share/hw5/* ~/cs170/hw5/

   Change your current directory to your newly created hw5 directory:

   cd ~/cs170/hw5/

3. You can now run gedit to edit your programs:

   gedit LookPrime.java &

   and

   gedit FindPositions.java &

Problems

1. LookPrime.java

   Prime numbers are numbers greater that can only be divided by themselves or by 1. Nobody has ever been able to find a formula for primes (the conjecture is that there is no formula). However, the following equation produces numbers that in many cases are primes: \( f(n) = n^2 + n + 41 \).

   For example, if \( n = 1 \), then \( f(1) = 43 \), which is prime. You can try it out with many natural numbers, and you will get prime numbers. The formula does not always work. For example, it clearly fails with \( n = 41 \). The purpose of this program will be to find examples in which the formula does not work. To do this, it should ask the user for a two positive integers \( a \) and \( b \) (say, 1 and 70), and the program will try this formula for the range between \( a \) and \( b \). Once it finds an example where the formula does not work, the program should print the number and the factors of
this number (which proves that it is in fact not a prime). The following is a sample run:

```
$ java LookPrime
Please provide a lower bound:
1
Please provide an upper bound:
70
The following are numbers with which the formula fails:
  1681 = 41 x 41
  1763 = 41 x 43
  2021 = 43 x 47
  2491 = 47 x 53
  3233 = 53 x 61
  4331 = 61 x 71
```

Hints: 1) Make a boolean method called isPrime to determine if a given input is a prime number or not. 2) Make an int method called factor so that, given a number that is not prime, it returns one of its factors. The initialization of these two methods has already been provided for you. 3) Run a loop from the lower bound to the upper bound. In each iteration, obtain the result of evaluating the function. Check if it is prime. If it is not, print the number and its factors. A method to print the factors has already been provided for you as well.

2. FindPositions.java
   Write a program that reads two strings as command line arguments. Check that there are two command line arguments and that the second string has unique characters. If it doesn’t have unique characters, write an appropriate message and exit (this has been done for you). Now, for each character in the first string, look for it in the second string, and print the position it is in. The output would therefore be a set of numbers of the same length as the first string stating the positions of the characters of the first string with respect to the second string. For example:

```
$ java FindPositions ddabdacde abdc
3 3 1 2 3 1 4 3 0
```
Notice that the second string has unique characters. Then take the first character in the first string (d), and look for it in the second string. Print 3, since d is in the third position. The next character is d again, so print 3 again. The next character is a, which appears in the first position of the second string, so print 1, and so on. If there is a character in the first string that does not appear in the second string, print 0. Such is the case of the last character, e, which does not appear in the second string. Here are a few more sample runs:

```bash
$ java FindPositions asdfdfaefa abadcef
The second string does not have unique characters!
```  
```bash
java FindPositions cerveza crze
1 4 2 0 4 3 0
```

Hints: Run a nested for loop. The first one for all the characters in the first string. Take the first character and look for it (with another loop) in the characters of the second string. If it appears, print the position (the indexes will tell you the position). If it went through all the characters in the second string and did not find it, then print 0 before entering again to the outer loop.

**Grading**

Your grade will be determined based on the correctness of your programs as well as program style. Program style includes such things as comments, variable/method names, and readability. Late penalties apply as per the class syllabus.

**Submission**

- When you finish, you should have 2 files in your /cs170/hw5 directory: LookPrime.java and FindPositions.java.
- Be sure to include the Honor Code statement at the top of EACH file you submit.
- Only your last submission will be graded.
- Submit your work using the following commands. You need to be in your /cs170/hw5 directory when you issue them:
Your homework is not turned unless the above commands are successful (you will get a "success" message when turn in was successful).