CS200: Programming I **Instructor:** Sam Vinitsky

P7: Prime Time!

(20 points)

Due: Thursday, July 18 @ 11:59pm

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's Jeff, local television personality! Jeff was recently hired to host the upcoming reality television show "Deserted", which strands 31 strangers on a deserted island. Each day, the participants compete in a challenge, and the loser must leave the island! On the last day, there will be one person remaining, who will win a whopping $100.

Jeff is beyond excited to be hosting this completely original and groundbreaking television program, but he needs your help! The challenges on the show are particularly complicated, and he isn't the brightest tool in the shed. That's why he's enlisted **you**! He has asked that you write two programs to help him prepare the challenges for the show.

Since Jeff will be running your programs while on a deserted island, his computer will be very weak. **It is strong enough to run functions, conditionals, loops, and recursion.** However, it cannot run any of the following: arrays, ArrayLists, String parsing, the "Math" library, or anything we havent explicitly covered in class. If you have questions about what you are/not allowed to use, please ask on Piazza.

Each program you turn in should include a comment at the top with (1) your full name, (2) your student ID number, (3) your netID, and (4) the name of anyone you discussed the homework with (excluding Sam and Alex).

Warning: This assignment is designed to challenge your critical thinking and problem solving skills. This will likely be the most difficult assignment so far. **Ask for help when you need it.**

As always: **start early, ask questions, and have fun!**

Part 1: Challenge #1 [7 points total]

The producers have planned all of the challenges for the show except for the first and the last, which they let Jeff plan. For the first challenge, Jeff has designed a race: the contestants must race across a pond filled with piranhas, and then eat an entire goose liver. When they finish the liver, they are given a random number between 1 and 1 million, and they must compute the **sum of the digits** of that number. The contestant to finish the race last loses, and is kicked off of the show.

a) Finding the sum (5 points)

Jeff already has the piranhas and goose liver taken care of, but he needs help with the "digit sum" portion of the challenge. Jeff needs you to write a program called `Challenge1.java` that contains a function named `computeDigitSum`. This function should take a positive integer[1] as input, and return the sum of its digits.[2] Your function's signature must be:

`public static int computeDigitSum(int n)`.

Make sure your function produces reasonable results for every `n` between 1 and 1 million. It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`computeDigitSum(10)` should return `1`
`computeDigitSum(11)` should return `2`
`computeDigitSum(154)` should return `10`
`computeDigitSum(111111)` should return `6`

You should be able to test your function relatively easily by computing the answer for several numbers by hand, and comparing the results to your function's output.

b) Printing them all (2 points)

In the main of `Challenge1.java`, print out the sum of the digits for every number from 1 to 1 million, formatted as follows:

```
The sum of the digits of 1 is: 1
The sum of the digits of 2 is: 2
...
The sum of the digits of 1000000 is: 1
```

Hint: *In order to print this out, you will need to call `computeDigitSum` repeatedly from main...*

[1]Do not worry about the case where the input is negative.

[2]**I highly recommend using recursion to compute this, but it is not required.** You may use loops if you desire (but it'll be nasty). However, you still cannot use Strings functions, arrays, ArrayLists, etc.

Part 2: Challenge #30 [10 points total]

Since challenge #1 was trivially easy for the contestants, Jeff decided to make things a little harder for the final challenge. For this challenge, each contestant will be suspended over a pit of crocodiles while blinded-folded. Jeff will throw baseballs at them, which they must bat away with a rubber chicken. If they get hit with a baseball, they will be given a random number between 2 and 1 million, and will need to compute the prime factorization of that number (i.e., write that number as a product of primes). The first person to get a factorization incorrect loses.

Jeff has the crocodiles, blind-folds, baseballs, and rubber chickens taken case of. But again, Jeff needs your help with the more mathematical side of the challenge. He's asked you to make a program called `Challenge30.java` and include in it the functions described in this section.

a) Is it prime? (2 points)

In order to produce the prime factorization of a number, we first need a way to to determine whether a number is prime (i.e. whether its only factors are 1 and itself). Write a function called `isPrime` that takes as input an integer greater than 1, and returns `true` if it is prime, and `false` otherwise. The function's signature must be as follows:

```
public static boolean isPrime(int n)
```

Make sure your function produces reasonable results for every `n` between 2 and 1 million.[3] It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`isPrime(2)` should return `true`
`isPrime(5)` should return `true`
`isPrime(7)` should return `true`
`isPrime(10)` should return `false`
`isPrime(15)` should return `false`
`isPrime(17)` should return `true`

Hint: *Try using loops.*

[3] When testing this function, it is acceptable to google "list of prime numbers" to check your answers...

b) Prime factorization (6 points)

Now we are ready to compute the prime factorization of a number! Write a function called `factor` that takes as input an integer `n` greater than 1, and returns a `String` containing the prime factorization of `n`. The factors should be listed in **increasing** order, with `*` between them (see the printout below for formatting). Note that `1` is **not** a prime number, and should not be included in the factorization. The signature of the function must be as follows:

```
public static String factor(int n)
```

Make sure your function produces reasonable results for every `n` between 2 and 1 million.[4] It is okay if your function is slow for larger numbers. Below is a **small sample** of the outputs you should expect from your function:

`factor(10)` should return the string `2*5`
`factor(11)` should return the string `11`
`factor(18)` should return the string `2*3*3`
`factor(20)` should return the string `2*2*5`
`factor(100)` should return the string `2*2*5*5`

Hint: *I recommend writing this function recursively, but it is not a requirement. Either way, you should use the function `isPrime` your wrote above...*

c) Printing them all (2 points)

In the `main` of `Challenge30.java`, print out the prime factorization of every number from 2 to 1 million, formatted as follows: (yes, there are missing lines...)

```
The factorization of 4 is: 2*2
The factorization of 5 is: 5
The factorization of 7 is: 7
The factorization of 8 is: 2*2*2
The factorization of 9 is: 3*3
The factorization of 10 is: 2*5
```

[4] When testing this function, it is acceptable to google "list of prime factorizations" to check your answers...

Part ∞: Feedback Form [3 points]

Fill out this feedback form **after you submit** this assignment. Completion of this will count towards your grade, but your responses themselves will not affect your grade in any way (so be honest!).

What to turn in

On Canvas, turn in a zip folder named `<your_net_id>_P7.zip` containing the files:

- `Challenge1.java` [7 points]
  - `int computeDigitSum(int n)` (5 points)
  - `main` (2 points)
- `Challenge30.java` [10 points]
  - `boolean isPrime(int n)` (2 points)
  - `String factor(int n)` (6 points)
  - `main` (2 points)

Also complete the feedback form. [3 points]