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

P1: Ahhh, Pirates!!!

(15 points)

Due: Monday, June 24 @ 11:59pm

Part 0: Introduction

Brrrring brriiiiiing the phone rings – it's the head of the Secret Army of Madison[1] (SAM for short). There have been reports of pirates *rampaging* across Madison, and SAM needs your help to stop them! SAM heard you were taking an introductory computer science course, and have decided that qualifies you to help! They need you to write a series of programs to help take down the dastardly pirates that threaten our very way of life. The fate of Madison is on your shoulders.

SAM is part of an elaborate bureaucracy, so they use very very old computers. The computers can run Java, but cannot make use of most of Java's functionality! Conveniently, **they can run anything we've covered in class**. Specifically, your code may **not** use functions other than `main`, conditionals, ("if" statements), loops ("for" or "while" statements), String parsing, or built-in Java libraries like Math or Random. If you have questions about what you are/not allowed to use, please ask on Piazza.

Tip: Before attempting this homework, work through all of the practice problems from this week's lectures (available on Canvas here).

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

Part 1: Assessing the damages [4 points total]

The first part of tackling any pirate infestation is determining how much money the city lost. Each business that got robbed by the pirates reported to SAM how much money they had before and after the robbery. The results are summarized in the table below.

[1] my lawyers have advised me to say that the Secret Army of Madison totally does not exist...

Store name	Money before the robbery	Money after the robbery
Fresh	$10,000.00	$4,376.02
Subway	$2,000.00	$2.99
QQ's	$827.69	$543.21
Chipotle	$188,203.98	$25,767.03
Epic	$2,000,000.00	$1,999,999.99

SAM has requested that you analyze this data by writing a series of programs, as described below.

a) Reporting loss (2 points)

SAM would like to know how much money each business **lost** due to the pirate problem. Write a program called `Loss.java` that prints out each store's name, along with how much money they lost. You should calculate the loss amount for each store, and store it in a variable.[2]. The order of the printout does not matter, but each store should be on its own line.

For example, if "Samopolis" had $200.41 before the robbery, and $100.00 after, it would have lost $100.41. If "Danielville" had $33.33 before the robbery, and $22.22 after, they would have lost $11.11. For this example, your program's output would look like:

```
Samopolis lost $100.41
Danielville lost $11.11
```

b) Average loss (2 points)

SAM would also like to know the *average*[3] amount lost by the businesses that were robbed. Write a Java program called `Average.java` that computes this, and **prints out** that amount. For example, if the average loss was $60.00, the program would output:

```
The average loss was $60.00
```

c) Significant loss (2 points)

SAM only cares about which stores lost a significant amount of money from the pirate onslaught. SAM considers a loss to be significant if it is more than 43.21876% of the business' starting amount. Write a program called `Significant.java` that determines which businesses had significant loss.

[2]This should be named something reasonable. For example, Fresh's variable might be named `freshLoss` or `amountLostByFresh`

[3]if you don't remember how to find the average of a set of numbers, this is a simple but effective explanation

In particular, your program should output the percentage of their money that each store lost. Then on a new line, output all of the stores that lost a significant amount (you may type the names of the stores that lost a significant amount by hand into your code).

For example, if "Samopolis" had lost 10.44%, and "Danielville" lost 68.68%, your program's output would look like:

```
Samopolis lost 10.44% of their money
Danielville lost 68.68% of their money

Stores that lost a significant amount: Danielville
```

Part 2: Cracking The Pirate Code [6 points total]

SAM has obtained copies of the pirates' internal correspondence with one another but, unfortunately for us, they communicate in some sort of code. In this section, you will help SAM decode some of the pirates' messages.

The pirates use a relatively rudimentary system to encode their messages – they use Java to convert their message letter by letter into integers, and then just send the integers to one another.

For example, if they wanted to send the word "Argh", they would run the following snippet to determine the encoded message to send:

```
int one = (int) 'A';
int two = (int) 'r';
int three = (int) 'g';
int four = (int) 'h';

System.out.println(one + "," + two + "," + three + "," + four);
```

This code outputs "`65,114,103,104`", and so the message the pirates would send would be "`65,114,103,104`".

a) Stopping the next pirate attack (2 points)

SAM has discovered top-secret pirate correspondence indicating where they are going to attack next. It is imperative that you translate it before the attack happens! The message was encoded using the technique described above.

The message is as follows: `83,116,97,114,114,114,98,117,99,107,115`.

Write a program called `Attack.java` that translates the pirate code, and prints out the location the pirates are going to attack (you can print out every letter on its own line, or the entire location on a single line).

b) Who is the pirate king? (2 points)

SAM has discovered correspondence that includes the name of the pirate king, and needs you to translate it, so they can find and arrest him! Unfortunately, the pirates are sneaky, and since you foiled their attack in part a have adopted a new code. This time, after converting the letter to an integer, they multiply that value by 3, and then add 21. For example, "Argh" would be encoded to `216,363,330,333` as in the snippet below.

```
int one = (int) 'A' * 3 + 21;
int two = (int) 'r' * 3 + 21;
int three = (int) 'g' * 3 + 21;
int four = (int) 'h' * 3 + 21;

System.out.println(one + "," + two + "," + three + "," + four);
```

The secret message that contains the pirate king's name is `237,354,354,342`. Write a program called `King.java` that translates the pirate code, and prints out the name of the pirate king (you can print out every letter on its own line, or the entire name on a single line).

c) Finding the pirate headquarters (2 points)

After arresting the pirate king, the pirates are on thin ice. They have started using another new code, in hopes that they can throw you off their scent. This code works as follows: each letter is translated to an integer as before, then is multiplied by its position in the word (starting at 1). Then that value is divided by 77.[4] For example, the word "Argh" would be encoded as `0.8441558441558441, 2.961038961038961, 4.012987012987013, 5.402597402597403` (produce by the snippet below). Notice that now the pirates' secret messages use `double`'s, not integers!

```
int one = (int) 'A' * 1 / 77.0;
int two = (int) 'r' * 2 / 77.0;
int three = (int) 'g' * 3 / 77.0;
int four = (int) 'h' * 4 / 77.0;

System.out.println(one + "," + two + "," + three + "," + four);
```

[4] 77 is the pirates' lucky number, since this looks like two pirate hooks side by side!

SAM has discovered the location of the pirates' headquarters, encoded as described above. If they can decode this, they will be able to take down the pirates once and for all!

The message is: `0.8701298701298701, 2.155844155844156, 1.9090909090909092, 2.6493506493506493, 3.116883116883117, 3.8181818181818183`

Write a program called `HQ.java` that translates the pirate code, and prints out the location of the pirate headquarters (you can print out every letter on its own line, or the entire location on a single line).

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>_P1.zip` containing the files:

- `Loss.java` [2 points]
- `Average.java` [2 points]
- `Significant.java` [2 points]
- `Attack.java` [2 points]
- `King.java` [2 points]
- `HQ.java` [2 points]

It is fine if you also include your `.class` files, but this is not required (it may just be easier to include them than to delete the files in order to zip up your folder).

Make sure to complete the feedback form as well! [3 points]
