This assignment is intended to refresh your memory of Java and prepare you for Data Structures this semester. In particular, you should be familiar with the following concepts:

- Data types
- Variable initialization
- Conditions - if statements
- Loops - while, for
- Methods - return, parameters, method overloading
- Fundamentals of objects; inheritance; encapsulation
- Array and index
- Simple recursion

And have the following basic programming skills:

- Debugging code using print statements/test cases/breaking points/etc.
- Decent skills in reading and writing Java code
- Reading and writing pseudo code; translating pseudo code into actual code
- Reading and finding information from official documentation (here)

If you are not comfortable with the above concepts or skills, or find this assignment to be very difficult, please speak to your instructor about transferring to COMP181 this semester.

All code in this project must follow the Java Code Style Guide.

Part I.

The following questions will require you to use Practice.java, MyString.java, MyIntArray.java, and Test.java. For each question, an implemented solution is provided for you. *This solution is incorrect.* Your job is to figure out the ways in which it is incorrect and fix it. For each question, you must:

- Write test cases (in Test.java) to figure out why the solution is incorrect
- In a comment above the method (in Practice.java), explain how the initial implementation is incorrect
- Fix the method in Practice.java

1) Write a method called `reverseOrder` that takes an `MyIntArray` object as its parameter and returns a new array in reserved order.

   `reverseOrder([1, 2, 3, 4, 5])` should return [5, 4, 3, 2, 1]
   `reverseOrder([])` should return []

2) Write a method called `findClumps` that takes a `MyIntArray` object as its parameter and returns the number of clumps found in the `MyIntArray`. A clump in an array is a series of 2 or more adjacent elements of the same values.

`findClumps([1, 2, 3, 3, 1])` should return 1 since the only clump is 3, 3
`findClumps([1, 1, 1, 1, 1])` should return 1 since the only clump is 1, 1, 1, 1, 1
`findClumps([1, 1, 2, 1, 1])` should return 2 since there are 2 series of adjacent 1s in the array.
`findClumps([1, 2, 2, 3, 5, 5, 1])` should return 2 since there are 2 clumps - 2, 2, and 5, 5.

3) Write a method, `removeMyString`, that takes two `MyString` objects, base and remove, as its parameters and returns a version of the base string where all the instances of the remove string have been removed. You may assume that the removed string has at least 1 character.

`removeMyString("Hello World", "ll")` should return "Heo World"
`removeMyString("Hello World", "o")` should return "Hell Wrld"
`removeMyString("Hello World", "aaa")` should return "Hello World"

4) Write a method (using recursion) called `countChar` that takes a `MyString` and a `character` as its parameters and returns the number of times that character appeared in the string. Assume that string and character are both case sensitive.

`countChar("ABABABababababa", 'a')` should return 4.
`countChar("zzyyaa", '2')` should return 0.

Part II.

In the following questions, you will write code from scratch. Suppose you are asked to implement several classes for a video game, Civilization X. Given the documentation of each class, implement them.

Please properly test all the methods of your classes to make sure they work in different scenarios.

1) Create a class called `Nation`.

Instance variables:
`String name, int population, Nation enemy, Nation ally`
Constructor:
The constructor should take the name and population.
Methods:
1. `public void setEnemy (Nation n)`

This method takes another Nation n as the input, and sets this Nation's enemy to the Nation n. You should also set Nation n's enemy to this nation.

2. `public void setAlly(Nation n)`

   This method takes another Nation n as the input, and sets this Nation's ally to the Nation n. You should also set Nation n's ally to this nation.

3. `public String toString()`

   This method returns a formatted String in the following format:

If the nation has no enemy, return:

```
Aztec has 2045 people. It has no enemy. Its ally is Mongolian.
```

If the nation has no ally, return:

```
Aztec has 2045 people. Its enemy is Mongolian. It has no ally.
```

If the nation has no ally and no enemy, return:

```
Aztec has 2045 people. It has no enemy. It has no ally.
```

Otherwise, return:

```
Aztec has 2045 people. Its enemy is Sumerian and its ally is
Mongolian.
```

4. `public void backstab()`

   This method flips your enemy and ally. However, it does not affect the status of your enemy and ally.

5. `public String getRelationship(Nation n)`

   This method returns a String "Enemy" if Nation n is your enemy, returns "ally" if n is your ally, else return "no relationship".

2) Now you are asked to create a FireNation that is a child class of Nation. You must use inheritance to create this class. Here is the documentation of FireNation:

Instance variables

`String uniqueUnit, int numOfUnits`

Constructor:

The constructor should take name, population, uniqueUnit and nuOfUnits.

Methods:

1. `public String toString()`

   This method returns a formatted String in the following format:

   If the nation has no enemy, return:

   `Aztec has 2045 people. It has no enemy. Its ally is Mongolian. It has 390 unique units of Eagle Warrior.`

   If the nation has no ally, return:

   `Aztec has 2045 people. Its enemy is Mongolian. It has no ally. It has 390 unique units of Eagle Warrior.`

   If the nation has no ally and no enemy, return:

   `Aztec has 2045 people. It has no enemy. It has no ally. It has 390 unique units of Eagle Warrior.`

   Otherwise, return:

   `Aztec has 2045 people. Its enemy is Sumerian and its ally is Mongolian. It has 390 unique units of Eagle Warrior.`