CSC 301 Sections 402, 411, 701, 710 Fall 2017
Homework Assignment 1
Due as specified on D2L

Directions: Please complete the problems below. The assignment will be graded on a scale of 0 to 4. Each problem is worth 1 point. I will provide code which calls each of your methods on a variety of test cases. You will receive full credit for a problem if your method works properly for all or most of the test method calls. You may receive ½ point for a problem if your code compiles and works in some cases. No partial credit will be given for code that does not compile. Templates for each problem can be found in the accompanying **`HW1.java`** file. In a few days, I will also provide a class whose main method tests your code.

As you work on this assignment, you may feel free to discuss it at a conceptual level with other students in the course. However, when you write your code, **you must work on your own.** As you write your code, you may ask for help from me and/or a CDM tutor, but **not** from other students in the course, friends, etc. Needless to say, you **may not** copy a solution from the Web. Copying code from another student or from the Web is plagiarism.

To complete this assignment, you will write 4 methods of the **`HW1`** class as specified below. For generic methods, you may assume that prior to any of the example code below, there is a statement which creates an appropriate **HW1** object, specifying the type of the variable **T.** For example:

```
HW1<Integer> hw1Int = new HW1<Integer>();
HW2<String> hw1Str = new HW1<String>();
```

Also, assume that the following variables have been defined:

```
Integer myPhone[] = {3, 1, 2, 3, 6, 2, 6, 1, 0, 6};
String hw1Topic[] = {"review", "of", "java", "language", "features"};
String csc301Topic[] = {"topics", "include", "balanced", "binary",
                        "search", "trees"};
```

1. Write a method called **`smallest`**. It is passed an array of **`T`** objects, and returns the "smallest" object in the array. If **`o1.compareTo(o2)`** is negative, then **o1** is smaller than **o2**. You must use a "for-each" loop to iterate through the array. Do **not** use any other types of loops in **`smallest`**. For example, the code on the next page should print

   ```
   0 features balanced
   ```

```
System.out.print(hw1Int.smallest(myPhone) + " ");
System.out.println(hw1Str.smallest(hw1Topic));
System.out.println(hw1Str.smallest(csc301Topic);
```

2. Write a method called **occurrences**. It is passed an item of type **T** and a Java List of type **T**. The method returns the number of times the item occurs in the List. For example, the code below should print **3 1 0**. For this problem, you must use a while loop instead of a "for-each" loop.

```
List<Integer> phoneList = Arrays.asList(myPhone);
List<String> hw1List = Arrays.asList(hw1Topic);
LinkedList<String> topicList = new LinkedList<String>();
for (String word : csc301Topic)
   topicList.add(word);
System.out.print(hw1Int.occurrences(6, phoneList));
System.out.print(" " + hw1Str.occurrences("java", hw1List));
System.out.println(" " + hw1Str.occurrences("programming", topicList));
```

3. Write a method called **subSequence.** It is passed two arrays, and determines if the items in the first array appear in the same sequence somewhere within the second array. For example, the following should print **true**:

```
Integer[] prefix = {3, 6, 2};
System.out.println(hw1Int.subSequence(prefix, myPhone));
```

On the other hand, this should print **false**:

```
String[] java = {"java", "features"};
System.out.println(hw1Str.subSequence(java, hw1Topic));
```

It is likely that you will use a nested loop structure to write this method.

4. Write a method called **powersOfN**. The method is intended to create an object which then be used to generate the sequence **1, n, n^2, n^3, n^{max}**. It is a **static** method, and is passed values for **n** and **max**. The method should return an object of type **Iterable<Integer>**. The **Iterable<T>** interface returns an object of type **Iterator<T>**, which in turn has methods called **hasNext** and **next**. Here is an example of how the **powersOfN** method would be used:

```
Iterable<Integer> p = HW1.powersOfN(10, 4);
for (Integer i : p) {
   System.out.print(i + " ");
System.out.println();
```

The above code should print **1 10 100 1000 10000.**