CS 323 (Spring 2018)
Assignment 2: Hashing with Separate Chaining
Due: 3/5/18 (11:59pm)

Problem and Requirements:

This programming assignment is based on (in part) Exercise C-10.32 from your textbooks.
In this project, you must implement a hash table and take care of collisions using separate chaining. The end product of this project is a Java class called `HashListMap` that implements a Map using hashing with a linked list in each hash bucket to handle collisions.

The following two interfaces will be given to you, and the assignment is to implement both of them:

- `public interface Map`
- `public interface Entry`

do not change these interface files.
To help get you started, you will be provided with two partially written Java classes that implement the above interfaces:

- `HashListEntry`
- `HashListMap`

In each of these files, there are a few methods missing. The first (and simpler) of your tasks is you must write those methods according to the specifications given in comment near the method definition. Once you have completed the implementation, you should have a fully functional hash table which handles collisions using separate chaining. However, the size of the table itself will be fixed, as there is no resizing method.

The second part of your programming assignment is to implement a resizing function. The standard resizing paradigm is to wait until the number of entries on the table exceeds half of the number of buckets, and then double the number of buckets. However, it is ideal for the efficiency of the hash function if the number of buckets is always prime. Therefore, you will create a method that implements the sieve algorithm outlined in exercise C-10.32 of your textbooks. If we let m be the number of buckets in your hash table, the ultimate goal is to find the first prime number that is greater than 2m.

Once you have written the method to locate the correct prime number, you will write a resizing method for your hash table that will, for a hash table with m buckets, resize the hash table to the size equal to the first prime number greater than $2m$. You will have to modify your `insert` method in order to call on the resizing method at the correct time, which is when the number of entries on the table exceeds half of the number of buckets.

Note: Your hash table does not have to resize if the number of entries shrinks.
Both of the provided classes has a `toString()` method written that prints the contents of the object. You will be given a test program `MyTestMap` which you can use to help debug your methods. The test program does a number of insert, get, and remove operations and prints the map after each operation.

Getting Started:

- Download the starter code in `/home/cs323001/share/hw1` and understand it.
- Fill in your implementation for the fixed size hash table.
- Run and test your program with various inputs. Use the test code that is provided to help debug, but try to alter the test code to test various cases that may occur.
- Write the method which returns the first prime number greater than $2m$ (where m is the number of buckets)
- Use the prime generating method to implement a resizable hash table.
- Run and test your program (again!) with various inputs. Use the test code that is provided to help debug, but try to alter the test code to test various cases that may occur.

Honor Code The assignment is governed by the College Honor Code and Departmental Policy. Please remember to have the following comment included at the top of the files.

```
/*
THIS CODE WAS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING
CODE WRITTEN BY OTHER STUDENTS OR SOURCES OUTSIDE OF THOSE
PROVIDED BY THE INSTRUCTOR.  _Your Name_Here_
*/
```

Submission:

Place your completed `HashListEntry.java` and `HashListMap.java` files directly under your `~/cs323/hw1` directory. Then use the turnin commands:

```
~cs323001/turnin HashListEntry.java hw2a
~cs323001/turnin HashListMap.java hw2b
```

Grading:

- If your program does not compile, you will get 0 points.
- Note that you must implement a hash table with separate chaining. If your program implements any other Map implementation, you will get 0 points.
- Correctness
  - Your program correctly inserts entries to the hash table when there is no collision (10pts)
  - Your program correctly handles collisions (10pts)
  - Your program correctly handles searching for and deleting entries with a given key. (20pts)
  - Your program resizes to the correct size (50pts)
- Code clarity and style (10pts)