Name: ______________________________ Andrew Id: ____________________

15-121 Sample Assessment 5

Up to 50 minutes. No calculators, no notes, no books, no computers. Show your work!

1. (10 points) **Hash Tables**

   Complete the following code for a hash table. When determining how many buckets you need, you should have 1.3x the expected number of items. Also, you need to ensure that duplicate items are never added to the table. (If there is an attempt to add a duplicate item, then do not add it.)

```java
public class MyHashtable<DataType> {
    private ArrayList<DataType>[] buckets;

    public MyHashtable(int numItems) {












    }

    public void add(DataType item) {












    }

    public boolean contains(DataType item) {












    }
}
```

2. **Buggy Sets**

Consider the following code for a Book class:

```java
public class Book implements Comparable<Book> {
    private String title;
    private String author;
    private int isbn;

    public Book(String title, String author, int isbn) {
        this.title = title;
        this.author = author;
        this.isbn = isbn;
    }

    // Natural order for books is defined by the title
    @Override
    public int compareTo(Book arg0) {
        return this.title.compareTo(arg0.title);
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        if (this.title != null) {
            result = prime * result + this.title.hashCode();
        }
        if (this.author != null) {
            result = prime * result + this.author.hashCode();
        }
        result = prime * result + this.isbn;
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null || !(obj instanceof Book)) {
            return false;
        }
        return this.compareTo((Book) obj) == 0; // ensure equals and compareTo are consistent
    }
}
```

After this code is deployed in a real product for a large library, the following bug report comes in:

> After we made Book objects for all two million books in the library, we added them all to a HashSet. Strangely, some of the books "disappeared" inside the HashSet. (Meaning that we called .add to add them to the HashSet, but they never actually got added.) In total, we've found that when we add all two million unique books to the HashSet, about 3 books disappear in this way.
>
> One of the librarians noticed that it seems like this only happens when two books have the same title, but different author. Strangely, though, it doesn't happen every time this is the case: Plenty of books with the same title and different author work just fine. Only three of them seem to disappear.

(a) (3 points) What is the bug in the code that causes this behavior?

(b) (4 points) Fix the bug in the code above. You can either modify the code directly on the previous page, or rewrite any methods that you need to here. (Note: There are multiple ways to fix the bug, but you should choose the way you think is best.)

(c) (3 points) Explain how/why the bug causes this behavior. Make sure your explanation covers why the bug happens in some cases, but not others, as mentioned in the bug report.

3. (20 points) **Free Response**

   In this problem you are going to write a Song class. Here are some important requirements for your class:

   - A Song has a title, artist, and runtime (in seconds). When a new Song is created, these items are passed to the constructor. (You must provide the constructor.)
   - The only way for code outside of Song to read and modify the attributes of a Song must be using appropriately named getters and setters. (You must provide the getters and setters.)
   - A title must not be null. If an attempt is made to set the title to something invalid, an IllegalArgumentException must be thrown.
   - An artist must not be null. If an attempt is made to set the artist to something invalid, an IllegalArgumentException must be thrown.
   - The artist may not be "Rick Astley". If an attempt is made to set the artist to "Rick Astley" then throw a generic exception with the message "No rickrolls today".
   - The runtime must be a positive integer. If an attempt is made to set the runtime to something invalid, an IllegalArgumentException must be thrown.
   - When printing a Song, its type and the value of its instance variables should appear. (Such as when System.out.println(someSongObjectHere) is used.)
   - A Song needs to be able to be properly used in HashSets and HashMaps.
   - An array of Songs must be sortable using Arrays.sort(someArrayOfMovies). The natural order is based on the title, with ties being broken first by the artist name and then by runtime.

   Write the Song class to the above specifications. Pay careful attention to which classes you may need to extend or implement as well as which methods you need to write or override.

Extra space for Question 3.