NAME:

CS 2102
Exam 1
D-Term 2017

Question 1:	_____	(40)
Question 2:	_____	(15)
Question 3:	_____	(25)
Question 4:	_____	(20)
TOTAL:	_____	(100)

If a problem asks you to create a class hierarchy, we are looking for the interfaces, classes, and abstract classes that you would create for the problem. In particular,

- Include **implements** and **extends** statements
- Include field names and types
- Include method headers (names, return type, and input parameter types)
- Share code where feasible
- Full credit requires that all types and implements/extends relationships are clear. Be sure your work is clear if you use UML diagrams instead of Java syntax.
- You may omit constructors
- You may omit method bodies
- You may omit the `Examples` class (examples of data and test cases) unless a question asks otherwise

1. (40 points) A library keeps different kinds of library items in its collection. You start with this class definition:

```
class Library{
  LinkedList<LibraryItem> items;
  ...
}
```

A LibraryItem can be a *book*, a *CD* (that is, a book recorded on CD), or a *magazine*. All library items have a title. Books and CDs also have an author. In addition, books have a boolean indicating whether or not the book binding is hard-cover (as opposed to paperback), and CDs keep track of the number of disks. In addition to a title, a magazine has two ints representing the volume number and issue number.

Library items are categorized as either *print* media or *electronic* media. Books and magazines are print media, while CDs are electronic media. All print media have a boolean method `isHardcover`. The method returns the value of the hard-cover field for a Book, and always returns false for a magazine.

Books and books on CD must have a method with the signature

```
boolean byAuthor (String author)
```

that returns true if the author of the book/CD is the same as the given author.

Provide whatever classes, abstract classes, and interfaces you need to capture this information (follow the guidelines on the front of the exam). You may provide your answers either by drawing a UML diagram or by writing Java code. You do NOT have to write constructors or the bodies of methods (use `{...}` to indicate the bodies of methods).

You may use this page and the next page for your answer.

(extra page for Problem 1)

2. (15 points) For each of these problems, fill in the "Named Values" and "Objects" sections of the memory map (you don't have to provide the "Known Classes" or "Expressions"). Both problems use these class definitions (constructors have been omitted to save space):

```
class Dog{
  String name;
  Toy favoriteToy;

  boolean hasRedToy(){
    return this.favoriteToy.color.equals("red");
  }
}

class Toy{
  String type;    // ball, frisbee, bone, etc.
  String color;
}

class Owner{
  String name;
  Dog pet;
}
```

(a) (5 points) Fill in the Named Values and Objects sections *after* these expressions have executed:

```
Dog fido = new Dog("Fido", new Toy("ball", "red"));
Owner karen = new Owner("Karen", fido);
```

```
KNOWN CLASSES       |   OBJECTS
                    |
                    |
                    |
                    |
                    |
                    |
                    |
                    |
                    |
                    |
--------------------------------------------------------------------
NAMED VALUES        |   EXPRESSIONS
                    |
                    |
                    |
```

(b) (10 points) Fill in the Named Values and Objects sections *during the execution of* the `hasRedToy()` method:

```
...new Owner("Jay",
             new Dog("Rex",
                  new Toy("bone","yellow"))).pet.hasRedToy()...
```

```
KNOWN CLASSES        |   OBJECTS
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
-------------------------------------------------------------------
NAMED VALUES         |   EXPRESSIONS
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
                     |
```

3. (25 points) Use this class definition for Problem 3 (constructor omitted):

```
class Student{
  String name;     // the student's name
  int yog;         // the student's expected year of graduation
  double gpa;      // the student's GPA (grade point average)
}
```

You're asked to design a method that satisfies this signature and purpose:

```
// Determines which year of graduation had the most students with
// a GPA of 4.0, for the students in studentList.
// baseYear is the earliest of four years represented in
//   studentList (see explanation below)
int yearWithMostTopGPAs(int baseYear,
                        LinkedList<Student> studentList);
```

You need to plan out the solution to `yearWithMostTopGPAs`.

You may assume that in `studentList`, there are at most four consecutive years of graduation represented, starting with the `baseYear`. For example, if the value of `baseYear` is 2010, then the years of graduation of Students in the list will be one of [2010, 2011, 2012, 2013]. You may not assume that Students in the list will appear in order of year of graduation, however.

If two or more years end in a tie for the most students with GPAs of 4.0, the method should return the earliest year. For example, if 2011 and 2013 both had 15 students with GPAs of 4.0, and the other years had fewer number of 4.0's, the method should return 2011.

(a) (10 points) Identify the subtasks for this problem

(b) (15 points) Suggest a way to solve the problem. You do NOT have to write any Java code; simply describe the steps you would take to solve the problem. Your descriptions might be worded as, "write a helper that...", "write a loop that...", "define a class that...", etc. Provide enough detail to convince us that you know how to solve the problem. Please provide your answer as a bulleted list rather than as one long paragraph.

4. (20 points) Use this class definition for Problem 4 (constructor omitted):

```
clas Song{
  String title;
  String artist;
}
```

(a) Write the Java code to implement this method:

```
// produces a list of the titles in songList that are by the given artist
LinkedList<String> titlesBy (LinkedList<Song> songList, String artist);
```

You must use an element-based for loop in your solution.

(b) Write the Java code for the `titlesBy` method again, but this time use an index-based for loop in your solution.

(extra page to use if you run out of room)