Due Date: July 28, 2002 11:59 pm
Group Size: 2
Submission Command: submit hw5
Weight: 7 points

Getting Started: Copy the files for Homework 5 from the directory $master/hw/hw5.

Exercises: As you complete the following exercises, place the code you write within the specified files. The files you copied contain framework necessary for this assignment. Do not change their structure except where instructed by comments in the framework.

1. (Inheritance) When writing a compiler, it is often useful to find the closest common parent class between two classes. Your task is to write code that, given a set of classes, some of which explicitly inherit from other class(es), and pairs of classes, output the closest common parent between the pairs of classes.

Each line of input will contain a class name by itself or a class name followed by the word inherits and then one or more whitespace-delimited classnames. Assume that all given classnames contain no embedded whitespace and are alphanumeric. If an input does not specify inherits, the class implicitly inherits from the class Object (although this may be explicitly specified). After a blank line of input, each line of input until end-of-file will contain class name pairs (a class name followed by whitespace and another class name).

On one line of the output you are to print the closest common parent in the inheritance hierarchy for each of the given class pairs. If X and Y are in a class pair and X is an ancestor of Y, output X. If two parents are of equal closeness, output the one of lower alphabetic value. If an inheritance circularity exists in the input, output only Circulariy found. and exit. Place your code in the file Inheritance.java

Sample input:

```
Foo
Bar
Foobaz inherits Foo
Foobar inherits Foo Bar

Foo Bar
Foo Foobar
Foobar Foobar
```

Sample output:

```
Object Foo Bar
```

2. (Hash Codes) For the classes below, fill in the appropriate methods in their files to allow their insertion into java.util.Hashtable such that the behavior of the hashtable is well-defined and fast (after many insertions and removals of any non-equal instances of a particular class, get should still be Θ(1)). You should use the instances of the below classes as the key and object, that is, given an instance of java.util.Hashtable, h, and an object of one of the types below, o, h.put(o, o) should place the item into the hashtable using the methods you wrote. Your hash functions should produce unique values for all distinct objects.

a. Instances of IntList1 are distinct based upon the items in the list and their positions.
b. Instances of Tree1 are distinct based upon the items in tree, but not the positions of those items.
c. Instances of Tree2 are distinct based upon the items in tree and their positions in the tree.
d. Instances of Matrix are distinct based upon the items in matrix and their positions in the matrix.