Birthday: Month: _____ Day: _____

Summer 2010
CS61BL Midterm 2

You have 110 minutes to finish this test. Your exam should contain 7 problems (numbered 0-6). This is an open-book test. You may consult any books, notes, or other paper-based inanimate objects available to you. Read the problems carefully. If you find it hard to understand a problem, please ask a question. Please write your answers in the spaces provided in the test; if you need to use the back of a page make sure to clearly tell us so on the front of the page.

Good Luck!

0		out of 1 point
1		out of 3 points
2		out of 3 points
3		out of 8 points
4		out of 6 points
5		out of 5 points
6		out of 6 points
Total		out of 32 points

Write your name on the last page

Birthday: Month: _____ Day: _____

Question 1: (3 points)

When you modify a key that has been inserted into a HashMap will you be able to retrieve that entry again?

☐ Always ☐ Sometimes ☐ Never

Explain:

When you modify a value that has been inserted into a HashMap will you be able to retrieve that entry again?

☐ Always ☐ Sometimes ☐ Never

Explain:

Birthday: Month: _____ Day: _____

Question 2 - Hashing (3 points)

Consider the following `Car class`. We plan to use `Car objects` as a `Key` in a `HashMap`.

```
public class Car {
      private int carNumber = 0;
      private static int carsCreated = 0;
      private String carName;
      public Car(String name){
            Car.carsCreated = Car.carsCreated + 1;
            this.carNumber = Car.carsCreated;
            this.carName = name;
      }
      public int hashCode(){
            return this.carNumber;
      }
      public boolean equals(Object obj) {
            Car otherCar = (Car) obj;
            return this.carName.equals(otherCar.carName);
      }
}
```

a) What are the strengths of the `Car class`? Write "None" if there are no strengths related to hashing.

b) What are the weaknesses of the `Car class`? Write "None" if there are no weaknesses related to hashing.

Question 3 (8 points):

You are provided the classes `Person`, `Address` and `AddressBook` in the handout. In the space on this and the next page, add code to any of the three classes or provide new versions of existing methods as necessary to implement the `getResidents` method for `AddressBook`. `getResidents` should return an `ArrayList` of `Person` objects that are associated with a given address. The `getResidents` method takes a single argument of an `Address` and must run in constant time. There are no restrictions on the time complexity of any other methods. java.util methods are outlined in the attached handout.

Please make it very clear in which class each method would appear.

More room for Question 3

Question 4 (6 points):

The handout contains the `BacteriaFamily` class that represents a tree. Answer questions below based upon the code provided in the handout.

a) Below please draw what memory looks like at the end of the `main` method provided in the handout. Note that the `insert` method calls a method `find`. You will be implementing `find` in part B, but if you have any questions about what it is supposed to do, please read ahead.

Question 4 continued:

b) Provide the body of the `static method find`, which will be called by the `non-static find method` shown in the handout. Your code should return the `Bacteria` object with the name `tryingToFind` if it is reachable from the node `current`, or return null if it cannot be found. You may only provide code for the `static method find` and may not change any other code.

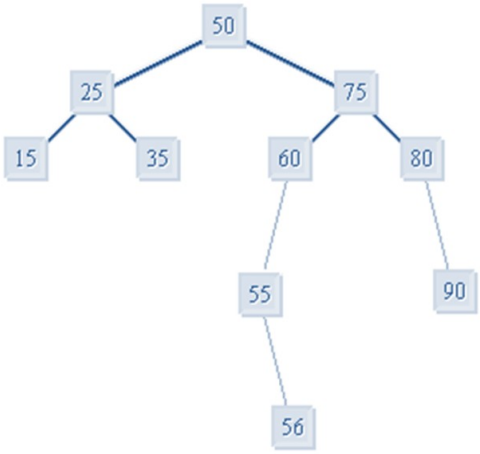
```
private static Bacteria find(String tryingToFind, Bacteria current) {
```

Birthday: Month: _____ Day: _____

Question 5: (5 points)

In the handout we have an abbreviated Binary Search Tree class called `BST`. The traditional `insert` and `remove` methods have been omitted for brevity. The `insert` method ignores attempts to insert elements already in the tree. The `remove` method will remove the inorder successor when necessary.

Node	pathFromRoot
15	"LL"
25	"L"
35	"LR"
50	""
55	"RLL"
56	"RLLR"
60	"RL"
75	"R"
80	"RR"
90	"RRR"



We added an instance variable to each node called `pathFromRoot`. We implement a method `paths` that stores for each node, a path from the root of L's and R's. The table above gives the value of `pathFromRoot` for each node in the tree on the right.

a) Assuming that `insert` is defined correctly for the BST class, after calling `insert`, would any node have a `pathFromRoot` that is not accurate? ☐ Yes ☐ No

If yes, please explain which nodes will have an inaccurate value for `pathFromRoot` and under what conditions. If no, please explain why not.

b) Assuming that `remove` is defined correctly for the BST class, after calling `remove`, would any node have a `pathFromRoot` that is not accurate? ☐ Yes ☐ No

If yes, please explain which nodes will have an inaccurate value for `pathFromRoot` and under what conditions. If no, please explain why not.

c) Complete the following sentence:
The runtime of the method `pathsPreorder` is proportional to

_____

d) Could we implement a more efficient algorithm for `pathsPreorder` that would improve the behavior of the runtime? ☐ Yes ☐ No
If yes, explain how this could be done. If not, explain why not.

Question 6:

The handout contains code for a class `IntList` and `IntListNode`. They are similar to the `List` and `ListNode` classes from lab. However, instead of a node holding a single value, an `IntListNode` object represents a range of numbers. In addition to a `next` and a `prev` pointer, `IntListNode` objects contain two instance variables, `min` and `max`, to represent the minimum and a maximum value for the range `[min, max]`. For example, an `IntListNode` with a `min` value of 3 and a `max` value of 6 represents the numbers 3, 4, 5, and 6 (or [3, 6]). You may assume that for each node, `max` is always greater than `min`. The method `reduce` combines any sequential `IntListNode` objects that represent sequential ranges of numbers. For example, the `IntList` below with 6 nodes would contain only 3 nodes after a call to `reduce`.

BEFORE `reduce()`: [3 , 5]→ [6 , 8]→ [9 , 12]→ [7 , 10]→ [5 , 9]→ [10 , 12]

AFTER `reduce()`: [3 , 12]→ [7 , 10]→ [5, 12]

a) Given the code in the handout, can the body of the method `reduceHelper` be written in each of these ways? Briefly explain.

Iterative:	Recursive:	Constructive:	Destructive:
☐ Yes ☐ No	☐ Yes ☐ No	☐ Yes ☐ No	☐ Yes ☐ No

Question 6 Continued

b) Write the non-static helper method `reduceHelper` for the `IntListNode` class. The method `reduceHelper`, is called by `reduce` in the `IntList` class. You may only write code in the `reduceHelper` method and may not change or add other methods.

```
private IntListNode reduceHelper() {
```

Page intentionally left blank.

You may use it for extra space. If you expect it to be graded you must indicate so clearly on the relevant problem.

Question #0

Write the month and day of your birthday on every page and complete the table below completely! (worth 1 pt)

Last Name:	
First Name:	
Login:	cs61bl-
TA (circle):	☐ 101 - Karan 8-11 ☐ 102 - Courtney 11-2 ☐ 103 - Stephanie 2-5 ☐ 104 - Eric 2-5
Birthday:	Month: _____ Day: _____