NAME: _____SAMPLE SOLUTIONS_____ NETID: _____XXXXXXX_____

CS2110 Fall 2009 Prelim 1
October 15, 2009

Write your name and Cornell netid . There are 9 questions on 9 numbered pages. Check now that you have all the pages. Write your answers in the boxes provided. Use the back of the pages for workspace. Ambiguous answers will be considered incorrect. The exam is closed book and closed notes. Do not begin until instructed. You have 90 minutes. Good luck!

	1	2	3	4	5	6	7	8	9	Total
Score	8/8	20/20	15/15	12/12	6/6	7/7	8/8	14/14	10/10	100/100
Grader	KPB	KPB	KPB	KPB	KPB	KPB	KPB	KPB	KPB	100

1. (8 points) Tell us what the following two Java programs will print.

```
class A {
        public static void main(String[] args) {
                A a = new A(); B b = new B(); A ab = new B();
                System.out.format("%d %d %d %d %d %d", a.x, b.x, ab.x,  a.y, b.y, ab.y);
        }
        int x = 2;
        int y = 100;
        A(int x) { this.x = x; }
        A() { this(3); }
}
class B extends A {
        int y = 200;
        B() { super(4); }
}
```

(4 pts) Output of A.main():

3 4 4 100 200 100

```
class D {
        public static void main(String[] args) {
                D d = new D(); E e = new E();
                System.out.format("%d %d", d.m(), e.m());
        }
        int m() { return (this instanceof E)? 5 : 6; }
}
class E extends D {
        int m() { return 100 + super.m(); }
}
```

(4 pts) Output of D.main():

6 105

2. (20 points) True or false?

a	T	F	In the Java class hierarchy, every class (except Object) extends exactly one parent class.
b	T	F	If during execution, an expression E is ever evaluated and its value is an object O, then the dynamic type of O is a subtype of the static type of E.
c	T	F	If we execute Integer a, b; a = b = new Integer(7); a = 11; b will still be equal to 7.
d	T	F	If class Dog includes a class variable "name" and we execute Dog a, b; a = b = new Dog("Biscuit"); b.name = "Rover", then a.name will also be changed to "Rover".
e	T	F	If class Dog includes a class variable "name" and we execute Dog a, b; a = b = new Dog("Biscuit"); b = new Dog("Rover"); then a.name will also be changed to "Rover".
f	T	F	If classes Cat and Turtle both implement the interface Pet, we can add objects of type Cat and objects of type Turtle to an ArrayList<Pet>.
g	T	F	If MyIterator implements the Iterator interface, we can add any object of type Iterator to an ArrayList<MyIterator>.
h	T	F	If the programmer does not specify a value for a **class variable**, the Java compiler automatically inserts code to initialize that variable to 0 or null, as appropriate.
i	T	F	If the programmer does not specify a value for a **local variable in a method**, the Java compiler automatically inserts code to initialize that variable to 0 or null, as appropriate.
j	T	F	If class Cat extends class Pet, and a new Cat instance is created, first the constructor for Cat will be called and then, automatically**, Java will call the constructor for Pet.**
k	T	F	If class Cat extends class Pet, and a new Cat instance is created, **the constructor for Cat should invoke super()** to ensure that the constructor for Pet has a chance to execute.
l	T	F	A class that implements an interface doesn't need to implement every single method. Unimplemented methods will automatically be generated by Java as "do-nothing" code.
m	T	F	If Pet is an interface, Java will not allow you to declare a variable to be of type Pet.
n	T	F	Examples of reference types include objects and arrays.
o	T	F	The arguments to a program are passed to the main(String[]) method as its arguments.
p	T	F	A class extends a single class but can implement many interfaces.
q	T	F	If a binary tree contains N nodes and is balanced, its maximum depth will be log(N)
r	T	F	Static methods may refer to static class variables but cannot refer to "this" or to dynamic class variables.
s	T	F	If a subclass overrides a method, the corresponding method in the parent class will not be executed.
t	T	F	In Java, a Container uses a LayoutManager to organize the layout of its Components.

3. (15 points) Consider the following grammar for simple arithmetic expressions.

<sae> ::= <num> | (<sae>) | <sae> + <sae> | <sae> - <sae> | <sae> * <sae> | <sae> / <sae>
<num> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

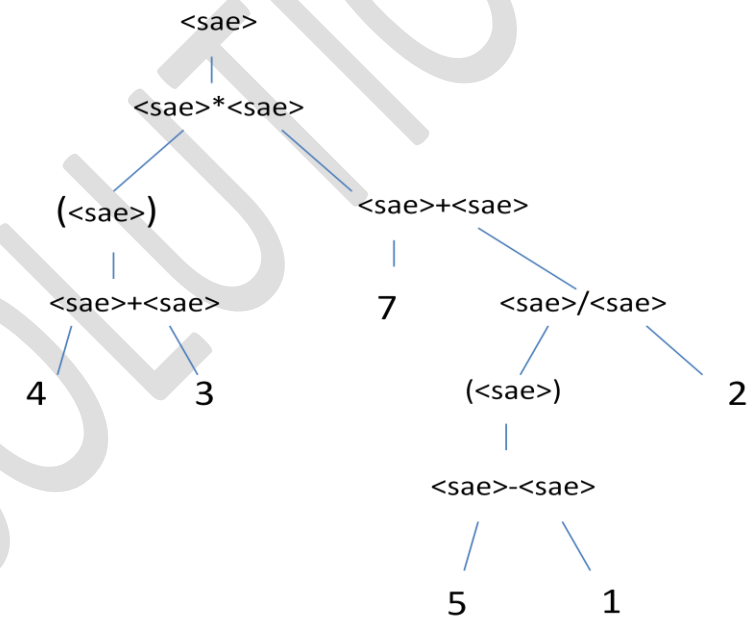
The start symbol is <sae>. Operators all have identical precedence and parsing occurs left to right.

(a) (1 point) How many sentences can be generated by this grammar? ***Infinitely many.***

(b) (7 points) For each of the following expressions, say whether it is a sentence generated by the grammar.

(i)	Y	N	(3 * 7) + (9 / 2)
(ii)	Y	N	((3) + (7)) / (2 + 1 + 8))
(iii)	Y	N	9 / 0
(iv)	Y	N	(9 + 2) (6 / 7)
(v)	Y	N	(((((8)))))
(vi)	Y	N	9.5 / 6.1
(vii)	Y	N	3

(c) (7 points) Draw the parse tree of (4 + 3) * 7 + (5 – 1) / 2



Note: We'll accept any tree "identical" to this one; you don't have to use exactly the same notation as we did to get full credit. But your tree does have to have the same nodes and structure because this grammar can only parse the given expression in this specific way!

4. (12 points) Choose the best alternative.

(a) “Autoboxing” refers to

A. the use of an anonymous class for an event handler.
B. the automatic conversion of a primitive type to its corresponding reference type.
C. the automatic conversion of a dynamic type to its corresponding static type.

(b) In Java, (x instanceof T)

A. is true only if x is an object of type T
B. is true only if x extends type T
C. is true if x implements the interface associated with type T (in effect, x can be “treated” as if it was of type T)

(c) An adapter is

A. a class that implements an interface by providing dummy implementations of all the interface methods.
B. an interface that specifies the mode of interaction between two classes.
C. a method that converts an object to a different type.

(d) A successful cast (Integer)x

A. changes the dynamic type of the object occupying x to Integer.
B. changes the static type of the variable x to Integer.
C. allows variable x to be treated as if it was of type Integer.

(e) If a line reads: “try { *stmt* } catch (Exception e) { }; “

A. no exceptions can arise while executing *stmt,*
B. an exception would interrupt execution of *stmt*, much like a break that breaks out of a loop, and execution will continue with the next line,
C. Java would flag this line as illegal because a catch clause can’t have an empty body.

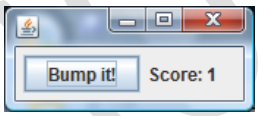
(f) In assignment 3, an object of type HashMap<String,ArrayList<Animal> > could be used:

A. To store the list of genes associated with a particular animal.
B. To maintain the animal-to-animal distance table.
C. To keep track of the animals that share a given gene.

5. (6 points) Here's some SWING GUI code similar to something we saw during Lecture 12.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Intro extends JFrame {
   private int score = 1;
   private JButton myButton = new JButton("Bump It!");
   private JLabel label = new JLabel("Score: " + score);
   public Intro() {
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(new FlowLayout(FlowLayout.LEFT)); //set layout manager
      add(myButton); //add components
      add(label);
      label.setPreferredSize(new Dimension(60, 10));
      myButton.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            score = score * 2;
            label.setText("Score: " + score);
         }
      });
      pack();
      setVisible(true);
   }
   public static void main(String[] args) {
      new Intro();
   }
}
```

(a) (2 points) Draw the frame that is displayed when the program is executed.



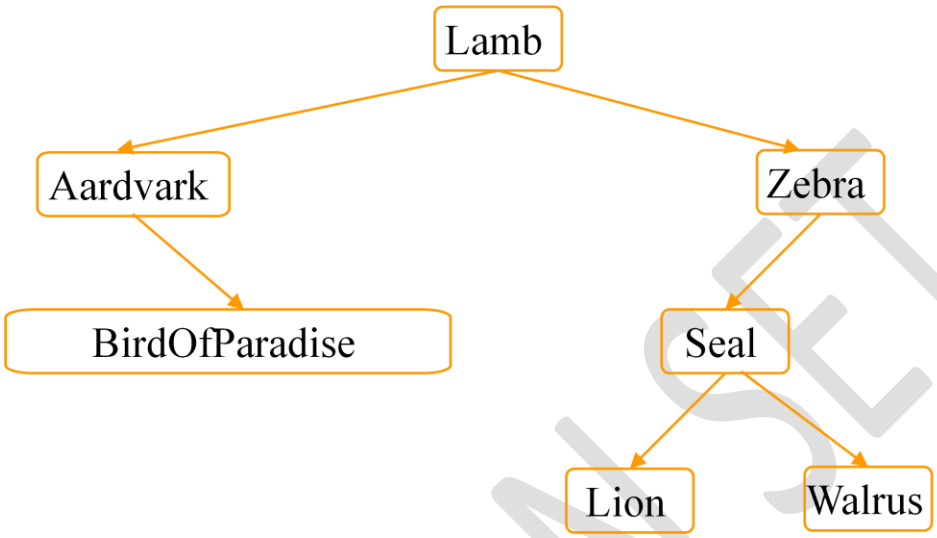
Note: We won't deduct if your picture lacks the stuff on the top line of the control (the cup of coffee, the minimize and maximize button, and the "X" for exit)

(b) (4 point) What happens when the user clicks on "Bump It!"? Try to give a "step by step" answer: "First, the user presses her mouse button…." Use terms we defined in class.

1. Java detects that the mouse button was clicked and generates a mouse click event
2. The event is delivered to the ActionListener associated with myButton
3. This causes **`actionPerformed`** to be invoked
4. **`actionPerformed`** doubles the existing score and redisplays it. The value was initially 1 and will thus become 2 the first time we click the button
5. **`actionPerformed`** returns and control goes back to the Java runtime system, which waits for something else to happen.

6. (7 points) Draw a binary search tree built from the following data, inserted in the order shown:

Lamb Zebra Seal Walrus Aardvark Lion BirdOfParadise.



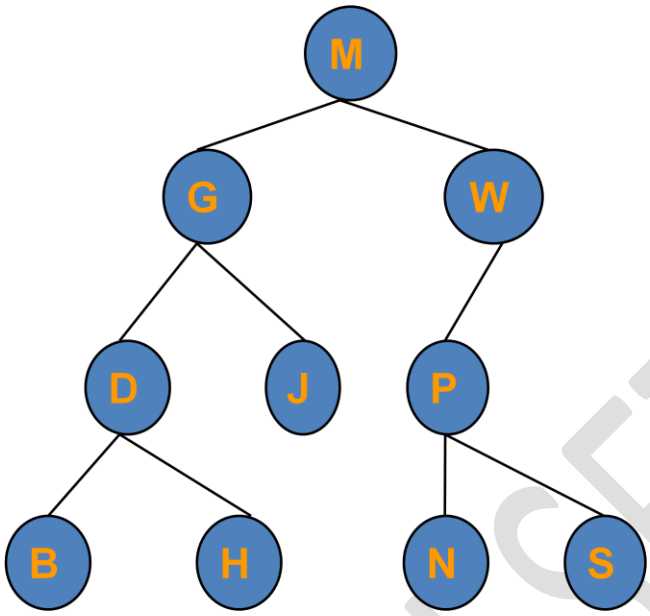
7. (8 points) Given a binary search tree on strings, write a **public ArrayList RangeSearch(String low, String high)** that returns a new ArrayList containing the values from the tree in the range [low…high] inclusive. The ArrayList should preserve the order of the values.

Hint: Popular ArrayList operations include "new" to allocate the list (it has no arguments), the "add" operation (appends to the end of the list), and the "contains" operation.

```
class BST {
    String value;          // Value of this node
    node left, right;      // Left, right children if any; else null

      public ArrayList RangeSearch(String low, String high) {
           ArrayList list = new ArrayList();
           RangeSearchWorker(list, low, high);
           return list;
      }

      private RangeSearchWorker(ArrayList list, String low, String high)
      {
          if(left != null) left.RangeSearchWorker(list, low, high);
          if(value.compareTo(low) >= 0 && value.compareTo(high) <= 0)
              list.add(value);
          if(right != null) right.RangeSearchWorker(list, low, high);
      }

}
```

Note: We didn't actually need the ArrayList "contains" operation.

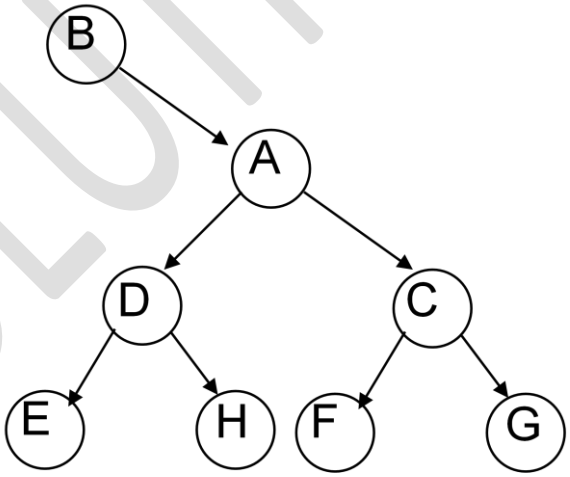
8. (14 points)
(a) (6 points) Write down the preorder, inorder, and postorder traversals of the following binary tree:



Preorder	M	G	D	B	H	J	W	P	N	S
Inorder	B	D	H	G	J	M	N	P	S	W
Postorder	B	H	D	J	G	N	S	P	W	M

(b) (8 points) Draw a (single) binary tree with nodes A–H having (both) the following traversals.

Preorder: B A D E H C F G
Inorder: B E D H A F C G



9. (10 points) Suppose that you are modifying a program that contains a LinkedList class implementing a constructor (creates an empty list), add(Object value) (adds a node) and contains(Object value) (returns true or false if the list contains or is lacking the value). Assume that our list contains a head node that has a null value and is used just so that even an empty list has at least one node on it. (Thus, if headNode is a list, and the list is *not* empty, the first node containing a value will be headNode.next). Write a method **public LinkedList deDup()** (called as **x = y.deDup()**). Given a list **y** that might contain duplicated nodes (with the same values), your method should return a new list of nodes in the same order but with duplicates removed. *Do not assume that list **y** is sorted.*

```
class LinkedList {
    Object value;    // For head node, this will always be null
    LinkedList next; // Next node or, if this is the last one, null
    . . . Not shown: code implementing the LinkedList constructor and methods . . .

    public LinkedList deDup() {
        new LinkedList list = new LinkedList();
        for(LinkedList lp = this.next; lp != null; lp = lp.next)
            if(list.contains(lp.value) == false)
                list.add(lp.value);
        return(list);
     }
}
```