Name: NetID:

Final exam

CS 2110, December 15, 2016, 9:00AM

Question	Short	Trie	Loop	Exp	Rec	Gen	Span	Mon	BigO	Data	**Total**
Max	20	7	9	10	9	9	8	10	10	8	100
Score											
Grader											

The exam is closed book and closed notes. Do not begin until instructed.

You have **150 minutes**. Good luck!

Write your name and Cornell **NetID** at the top of **every** page! There are 10 questions on 14 numbered pages, front and back. Check that you have all the pages. When you hand in your exam, make sure your pages are still stapled together. If not, please use our stapler to reattach all your pages!

We have scrap paper available. If you do a lot of crossing out and rewriting, you might want to write code on scrap paper first and then copy it to the exam so that we can make sense of what you handed in.

Write your answers in the space provided. Ambiguous answers will be considered incorrect. You should be able to fit your answers easily into the space provided.

In some places, we have abbreviated or condensed code to reduce the number of pages that must be printed for the exam. In others, code has been obfuscated to make the problem more difficult. This does not mean that it's good style.

Academic Integrity Statement: I pledge that I have neither given nor received any unauthorized aid on this exam.

_____

(signature)

1. Short Questions [20 pts]

(a) [3 pts] **Hash tables.** Consider hashing using linear probing. Write the definition of *load factor*.

(b) [4 pts] **Exceptions.**
Consider the method given below. Suppose the try-block throws an ArithmeticException, so the catch-block is executed. Write down how execution proceeds in two cases (1) The catch-block throws another ArithmeticException and (2) the catch-block does not throw another exception.

```
public void m() {
    try { ... }
    catch (ArithmeticException e) { ... }
    ...
}
```

(c) [3 pts] **Local variables.**
Consider method `m` declared below. To the right of the method, write down when during execution of the call `m(5)` space is allocated for variable `x`.

```
public static int m(int p) {
    p= Math.abs(p);
    while (p > 0) {
        int x= 2*p;
        return x;
    }
}
```

(d) [3 pts] **Function equals.** Let C be a class. The fields of an object of class C are not changed except by the constructor. Describe a situation in which the following expression is false.

```
(new C()).equals(new C())
```

(e) [3 pts] **Generics.** Give an example of what could go wrong if Java allowed LinkedList<Integer> to be a subtype of LinkedList<Object>

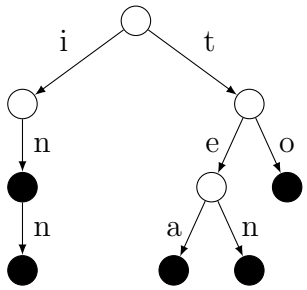
(f) [4 pts] **Constructors.**

- Write the constructor that Java inserts in class C if one is not declared in C.

- Write the statement that Java inserts as the first statement of a constructor if the first statement is not a call on another constructor.

2. Tries vs BSTs [7 pts]

The trie[1] data structure is a tree that maintains a sorted set of strings. Each edge in a trie is labeled with a single character. To look up a string, begin at the root and follow the path of characters in the string. Each node contains a boolean value to indicate whether the path from the root to that node represents a complete string that is in the set, and not just a partial prefix. This allows a word that is a substring of another to be in the set, for example, "in" and "inn" shown below. Here, the black nodes represent words in the set `{"in", "inn", "tea", "ten", "to"}`.



Assume that words can contain only the 26 letters a..z. Assume that the child of a node corresponding to a letter can be referenced in constant time (the child is either a node or null, in case the child is empty).

(a) [4 pts] **Complexity of tries.** Consider a trie with n strings of maximum length m. What is the tightest Big-O worst-case complexity (in terms of n and m) of determining whether the trie contains a given word?

(b) [3 pts] **Complexity of a BST containing strings.** Assume strings use only the characters in a..z, as in the previous question. Consider a balanced BST containing n strings of maximum length m. What is the tightest Big-O worst-case complexity (in terms of n and m) of determining whether the BST contains a given word?

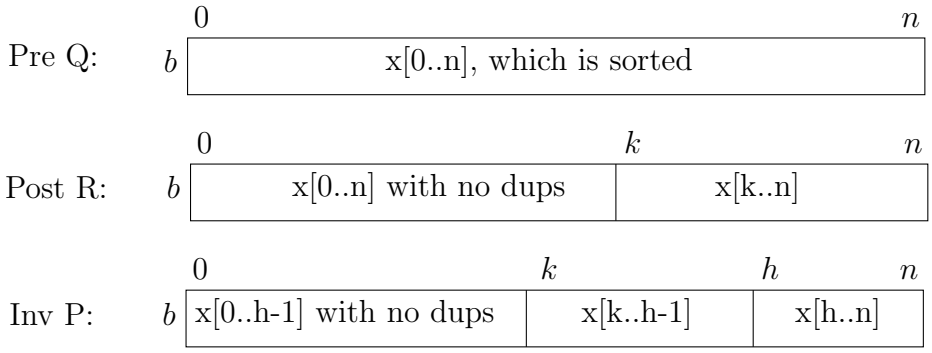
[1] from re*trie*val, a pun on tree

3. Loops [9 pts]

Given is `n > 0` and array segment `b[0..n]`, which contains a sorted list of integers, possibly containing duplicates. We want to place the *set* of integers in `b` at the beginning of `b`; the rest of `b[0..n]` remains unchanged. For example, below is one possibility for `b[0..n]`, with 5 distinct values, and below it is the result of the algorithm. At the end, variable `k` will contain the size of the set.

```
[2, 5, 5, 5, 7, 7, 8, 8, 9]
[2, 5, 7, 8, 9, 7, 8, 8, 9]      k = 5
 -------------  ----------
   set           unchanged
```

Here are the precondition, postcondition, and an invariant derived from them, using `x[0..n]` to denote the original list of values in `b[0..n]`. *dups* stands for *duplicates*.



(a) [2 pts] **Initialization**

To the right, write the initialization that truthifies the invariant.

(b) [2 pts] **Termination**

To the right, write the condition `B` such that `B` and `P` imply `R`.
Do not write "while" and things like that. Just write the condition.

(c) [1 pt] **Progress**

To the right, write a statement that makes progress toward termination.

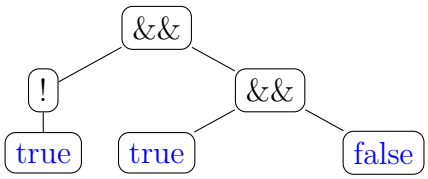
(d) [4 pts] **Maintain invariant**

To the right, write a statement that, when executed before the statement written in c), will ensure that the invariant is true after the statement in (c) is executed.

4. Expressions [10 pts]

To represent expressions, you have built the classes listed on page 7, which you may remove from the exam booklet. For simplicity, we are not representing || expressions.

(a) [3 pts] Write a Java expression using these classes that represents this tree:



Expr e = new

(b) [7 pts] Write a visitor class that, when applied, returns a String containing a preorder traversal of an expression. For example, e.apply(new PreorderVisitor()) should return "&& ! T && T F " (where e is the expression from part (a)). Hint: making the code below compile will get you started.

```
public class PreorderVisitor implements ExprVisitor<String> {




}
```

```
/** Represents a boolean expression. */
interface Expr {
    /** apply the visitor to this object and return what it returns. */
    <R> R apply (ExprVisitor<R> visitor);
}

/** Represents an operation to be performed on an Expr
 *  @param R is the return type of the operation
 */
interface ExprVisitor<R> {
    public R visitAnd(And a);
    public R visitNot(Not e);
    public R visitConstant(Constant c);
}

/** Represents the expression "left && right" */
class And implements Expr {
    public Expr left, right;
    public And(Expr left, Expr right) { this.left = left; this.right = right; }
    public @Override <R> R apply(ExprVisitor<R> visitor) {
        return visitor.visitAnd(this);
    }
}

/** Represents the expression '! child' */
class Not implements Expr {
    public Expr child;
    public Not(Expr child) { this.child = child; }
    public @Override <R> R apply(ExprVisitor<R> visitor) {
        return visitor.visitNot(this);
    }
}

/** Represents the expression 'T' or 'F' */
class Constant implements Expr {
    public boolean value; // true for 'T'; false for 'F'
    public Constant(boolean value) { this.value = value; }
    public @Override <R> R apply(ExprVisitor<R> visitor) {
        return visitor.visitConstant(this);
    }
}
```

This page intentionally left blank

5. Recursion [9 pts]

Consider a binary search tree whose values are integers and whose nodes are objects of class BST:

```
public class BST {
    private int value; // Value in this node
    private BST left;  // Left child, or null if none
    private BST right; // Right child, or null if none
    // additional code for methods add, lookup, etc.
```

(a) [3 pts] Complete function size:

```
/** Return the number of nodes in t (0 if t is null). */
public static int size(BST t) {




}
```

(b) [6 pts] Complete function numBigger, doing only as much computation as necessary:

```
/** Return the number of values in t
  * that are greater than v (0 if t is null). */
public static int numBigger(BST t, int v) {




}
```

6. Generics [9 pts] Consider the following class hierarchy:

```
interface Flier { void fly(); }

abstract class Animal {
    void     getWeight() { ... }
    abstract String getName();
}

abstract class Bird extends Animal
                implements Flier { }
```

```
interface List<E> {
    E get(int i);
    void add(E e);
    int size();
}


class Mammal extends Animal { ... }
class Dog    extends Mammal { ... }
class Parrot extends Bird { ... }
```

(a) [2 pts] What methods must Parrot implement?

(b) [1 pt] What methods must Dog implement?

(c) [3 pts] What is the most general type for flock in the following method (i.e. the one that will allow it to be called on as many different types as possible but still compile).

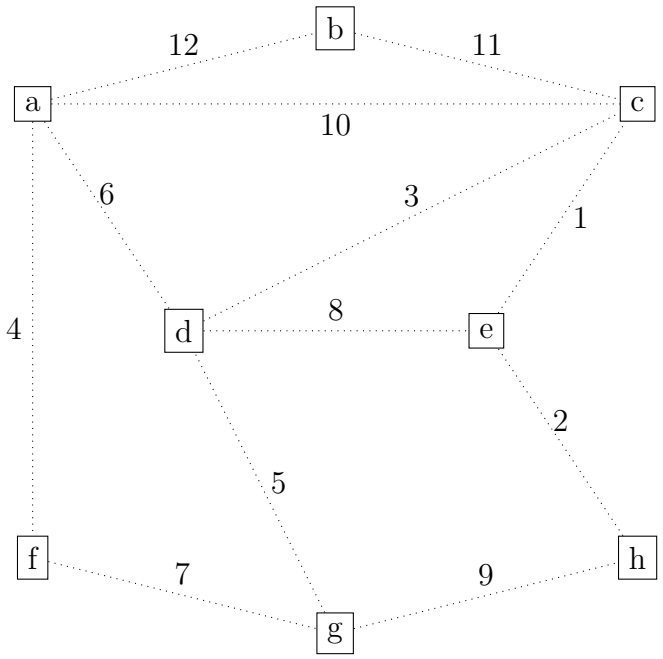
```
public void migrate(List<_____> flock) {
    for (int i = 0; i < flock.size(); i++)
        flock.get(i).fly();
}
```

(d) [3 pts] What is the most general type for herd in the following method?

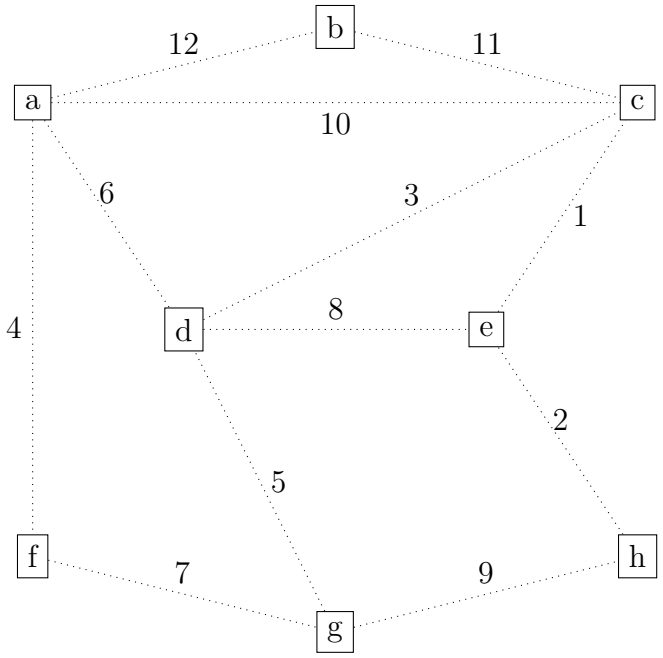
```
public void populate(List<_____> herd) {
    herd.add(new Mammal());
}
```

7. Graphs [8 pts]

(a) [4 pts] Execute Kruskal's algorithm on the following graph. Fill in the edges as you select them and number them: write ① next to the first edge you select, ② next to the second, and so on. If the algorithm requires a starting node, use a.



(b) [4 pts] Similarly, execute Prim's algorithm on the following graph. If the algorithm requires a starting node, use a.



8. Monitors [10 pts]

Complete the following implementation of a Monitor, which maintains two counters:

```
/** An XYMonitor maintains two counters, x and y, and ensures that x < y. */
class XYMonitor {

    /* FILL IN
     *
     */

    private int x, y;

    /** Constructor: an instance with counters x and y. Precondition: x < y */
    public XYMonitor(int x, int y) { this.x = x; this.y = y; }

    /** Block until it is safe to increment counter x, then increment x */
    public synchronized void incrementX() throws InterruptedException {
        // FILL IN



    }

    /** Decrement counter x. */
    public synchronized void decrementX() {
        // FILL IN



    }

    /** Increment counter y. */
    public synchronized void incrementY() {
        // FILL IN



    }

    /** Block until it is safe to decrement counter y, then decrement y. */
    public synchronized void decrementY() throws InterruptedException {
        // FILL IN



    }
}
```

9. Complexity [10 pts] Consider the worst-case execution of the algorithm below on a graph with N vertices and E edges.

   (a) [6 pts] Next to each line, write an asymptotic bound for the total time it takes to execute that line, assuming first that G is stored as an adjacency list and second that it is stored as an adjacency matrix. Write your answers in terms of E and N. Assume that each line is implemented as efficiently as is possible. The first two lines are completed for you.

total asymptotic run time (adj. list)	total asymptotic run time (adj. matrix)	
$O(N)$	$O(N)$	for each vertex v
$O(N)$	$O(N)$	print v
		copy the vertices into an array b
		sort b
		for each entry u in the array
		print u
		for each neighbor v of u
		find the index i of v in b
		print i

   (b) [2 pts] What is the total asymptotic running time for each?

   Adjacency list: Adjacency matrix:

   (c) [2 pts] Based on your answer to (b), which representation will cause the algorithm to run faster on a particular graph with 100 vertices and 8000 edges?

   i. The adjacency matrix representation will run faster
   ii. The adjacency list representation will run faster
   iii. There is not enough information to answer

   Explain your answer in 1-2 sentences.

10. Data structures [8 pts]

Identify which of the following data structures

A. Adjacency list
B. Adjacency matrix
C. Min heap
D. Singly linked list
E. Doubly linked list
F. Array
G. Hash table
H. (Balanced) binary search tree

is best for storing the following kinds of data:

a. ____ A contact list for looking up your friends' phone numbers
b. ____ Players waiting for their turn to start a game, who may get tired and leave
c. ____ Reminders that are displayed at a specified time
d. ____ Print jobs that are printed in the order they are submitted
e. ____ Cached results from a function that computes the Fibonacci numbers
f. ____ A contact list for displaying the name of someone who is calling you
g. ____ Web pages and the links between them
h. ____ Permissions for determining whether two users are allowed to speak with each other

Note: each data structure is matched with exactly one scenario.