University of Toronto
**CSC 148—Introduction to Computer Science, Spring 1999**

# Second Term Test
# Horton's and Reiter's Sections

**Duration:** 50 minutes

**Aids allowed:** None

Make sure that your examination booklet has 6 pages (including this one). Write your answers in the spaces provided. Write legibly.

**Family Name:** ______________________ **Lecturer:** ______________________

**First Name:** ______________________ **Tutor:** ______________________

**Student # :** ______________________

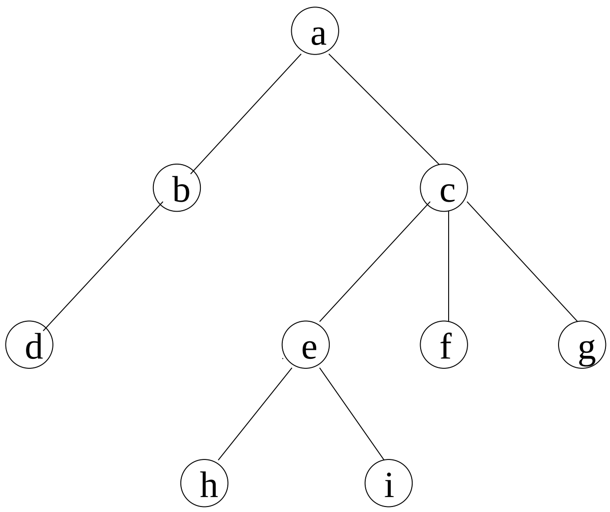
| | | |
|---|---|---|
| 1. | ________ | / 6 |
| 2. | ________ | / 10 |
| 3. | ________ | / 13 |
| 4. | ________ | / 8 |
| Total | ________ | / 37 |

## Question 1

[6 marks in total]

1. [2 marks] What properties must a tree have to be a Binary Search tree?
   (a) It must be binary, meaning every node has at most two children.
   (b) If a node has a left subtree, the values of all nodes in this subtree must be less than the node's value. ("less than or equal" is OK.)
   (c) If a node has a right subtree, the values of all nodes in this subtree must be greater than the node's value. ("greater than or equal" is OK.)

2. [4 marks] Consider the following tree:




Show the sequence of node labels produced by printing all the tree's nodes in:

**Preorder:** *a b d c e h i f g*

**Postorder:** *d b h i e f g c a*

## Question 2

[10 marks in total]

Consider the following class declarations:

```
class M{                                  |   class P extends M {
  String type;                            |
                                          |     public boolean le() {
  public boolean le() {                   |       return true;
    return false;                         |     }
  }                                       |   }
                                          |
  public boolean wb() {                   |   class Q extends D {
    return true;                          |     boolean hai;
  }                                       |
}                                         |     public boolean ish() {
                                          |       return hai;
class D extends M {                       |     }
                                          |   }
  public String name() {                  |
    return type;                          |
  }                                       |
}                                         |
```

In the spaces provided below, show what would be printed. It is possible that one or more of the statements would cause an error, either run-time or compile-time. In such cases, just write "error". If you are uncertain whether it is legal to print a `boolean` in Java, rest assured that it is.

```
public class Main {
  public static void main(String[] args) {

    D d = new D();
    d.type = "zip";
    System.out.println(d.wb());              Answer: true
    System.out.println(d.hai);               Answer: error
    System.out.println(d.le());              Answer: false
    System.out.println(d.name());            Answer: zip
    System.out.println(d.ish());             Answer: error

    Q q = new Q();
    q.type = "code";
    System.out.println(q.le());              Answer: false
    System.out.println(q.name());            Answer: code
    System.out.println(q instanceof Q);      Answer: true

    System.out.println((new P()).le());      Answer: true
    System.out.println((new P()).wb());      Answer: true
  }
}
```

## Question 3

[13 marks in total]

Consider the following method:

```
public int f(int x, int y) {
  if (x == 0)
     return y;
  else
     return f(x-1,y+1);
}
```

1. (2 marks) For what values of `x` and `y` does `f(x,y)` terminate? No proof is required of your claim.

   $x \geq 0$ and for all values of $y$.

2. (3 marks) Give a precise description of what `f(x,y)` computes.

   $x + y$.

3. (8 marks) Consider the following method:

   ```
   public boolean p(int n) {
     if (n == 0)
        return true;
     else if (n == 1)
        return false;
     else
        return p(n - 2);
   }
   ```

   Prove that `p(n)` terminates whenever $n \geq 0$, and that the value returned is `true` when `n` is even, and `false` when `n` is odd.

The proof is by induction on $n$.
Let $S(n)$ be the statement: For every integer $n \geq 0$, `p(n)` terminates, and returns `true` when `n` is even, and `false` when `n` is odd.

**First Base Case:** $n = 0$. Then `p(0)` terminates, and returns `true`. Since 0 is an even integer, $S(0)$ is true.

**Second Base Case:** $n = 1$. Then `p(1)` terminates, and returns `false`. Since 1 is an odd integer,

$S(1)$ is true.

**Strong Induction Hypothesis:** Let $k \geq 1$, and assume, for $i = 0, 1, \ldots, k$ that $S(i)$ is true.

**Induction Step:** We prove that $S(k+1)$ is true. Consider the call `p(k + 1)`. Now $k \geq 1$; therefore, $k + 1 \geq 2$, so the call `p(k + 1)` returns the value of the call `p(k + 1 - 2)`, which is the call `p(k - 1)`. By the induction hypothesis, the call `p(k - 1)` terminates, and returns `true` when $k-1$ is even, and `false` when $k-1$ is odd. Since $k+1$ is even when $k-1$ is even, and $k+1$ is odd when $k-1$ is odd, the call `p(k + 1)` terminates, and returns `true` when $k+1$ is even, and `false` when $k+1$ is odd, i.e. $S(k+1)$ is true.

**Alternative (Simple) Induction Hypothesis:** Let $k \geq 0$, and assume $S(k)$ is true.

**Induction Step:** We prove that $S(k+2)$ is true. Consider the call `p(k + 2)`. Now $k \geq 0$; therefore, $k + 2 \geq 2$, so the call `p(k + 2)` returns the value of the call `p(k + 2 - 2)`, which is the call `p(k)`. By the induction hypothesis, the call `p(k)` terminates, and returns `true` when $k$ is even, and `false` when $k$ is odd. Since $k+2$ is even when $k$ is even, and $k+2$ is odd when $k$ is odd, the call `p(k + 2)` terminates, and returns `true` when $k+2$ is even, and `false` when $k+2$ is odd, i.e. $S(k+2)$ is true.

## Question 4

[8 marks in total]

Suppose you have already defined an IntNode class that can be used for linked lists of `int`s, as follows:

```
class IntNode {
  public int value;
  public IntNode next;
}
```

Write a **recursive** body for the following method signature:

```
public int getLast(IntNode list) {
```

`getLast` returns `0` if `list` is the empty linked list, and otherwise it returns the value of the last node of the linked list `list`.

```
  if (list == null)
     return 0;
  else if (list.next == null)
     return list.value;
  else
     return getLast(list.next);
  }
```

It's possible to do this with a helper:

```
  if (list == null)
     return 0;
  else
     return help(list);  \\ list is non-empty.
  }

private int help(IntNode list) {
   if (list.next == null)
     return list.value;
  else
     return help(list.next);
  }
```