UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL EXAMINATIONS 1999

CSC 148S
St. George Campus

Duration — 3 hours

Examination Aids: One $8\frac{1}{2}$" $\times$ 11" sheet of paper, *hand*written on both sides.

First Name: _____

Last Name: _____

Student # : _____

Section: (check one)

- ☐ L5101 (evening) *Steve Bellantoni*
- ☐ L0101 (morning) *Paul Gries*
- ☐ L0201 (afternoon) *Diane Horton*
- ☐ L0202 (afternoon) *Ray Reiter*

Do **not** turn this page until you have received the signal to start.
(In the meantime, please fill out the identification section above, and *carefully* read the instructions below.)

This examination consists of 10 questions on 16 pages. *When you receive the signal to start, please make sure that your copy of the examination is complete.* Answer each question directly on the exam paper, in the space provided, and *use the reverse side of the pages for rough work.* (If you need more space for one of your solutions, use the reverse side of the page and indicate **clearly** which part of your work should be marked.)

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero (0) so write legibly. In your answers, you may use any theorems or facts given during the course, *as long as you state them clearly*. You must prove any other theorems or facts needed for your solution.

General hint: We were careful to leave ample space on the exam paper to answer each question. If you find yourself using much more room than what is available, you're probably missing something.

# 1:	_____/12
# 2:	_____/10
# 3:	_____/13
# 4:	_____/ 6
# 5:	_____/12
# 6:	_____/ 8
# 7:	_____/ 6
# 8:	_____/14
# 9:	_____/ 6
# 10:	_____/ 8
TOTAL:	_____/95

Good Luck! PLEASE HAND IN

1. [12 marks]

(a) [2 marks]

Print your name and student number *legibly* at the top of pages 2–16 of this test.

(b) [2 marks]

Circle `true` if the following statement is true, `false` otherwise. You will receive 2 marks for the correct answer, and -2 marks if you circle the incorrect answer.

> Using the fastest available algorithm, searching in a sorted doubly-linked list of n items is always faster than in a sorted array of n items, because the `prev` pointers allow us to search from either end of the list.

`true` `false`

Answer:
false

(c) [2 marks]

Circle `true` if the following statement is true, `false` otherwise. You will receive 2 marks for the correct answer, and -2 marks if you circle the incorrect answer.

> Searching in a binary search tree of n items is always as fast as a binary search in a sorted array of n items.

`true` `false`

Answer:
false

(d) [3 marks]

Consider a recursive method for binary search in an array. If you were to prove this method correct, would you use strong induction or weak induction? Circle one answer.

`Strong induction` `Weak induction`

Explain why:

Answer:
Strong induction, because the size of the subsection in the recursive call is half as big. (Also okay: because the "problem size" is half as big.)

(e) [3 marks]

Consider a recursive method for printing the elements of a linked list in reverse order. If you were to prove this method correct, would you use strong induction or weak induction? Circle one answer.

`Strong induction` `Weak induction`

Explain why:

Answer:
Weak induction. The recursive call is on a list one smaller. (Also okay: because the problem size is reduced by one.)

2. [10 marks]

For this question, you will examine a series of possible structures for proof by induction. $S(n)$ is some statement that is defined for all $n \geq 0$.

(a) [2 marks]

The following proof structure is not valid. Correct it.

BASE CASE(S): Prove that $S(0)$ is true.
Let k be an arbitrary integer ≥ 1.
Answer: This needs to be ≥ 0 to make the inductive step hook up with the base case.

INDUCTION HYPOTHESIS: Assume that $S(k)$ is true.
INDUCTION STEP: Prove that $S(k+1)$ is true.
CONCLUSION: $S(n)$ is true for all $n \geq 0$.

(b) [2 marks]

Fill in the blanks to make the following proof structure valid.

BASE CASE(S): Prove that _____.
Answer: $S(0)$ is true
Let k be an arbitrary integer $\geq$ Answer: 1.

INDUCTION HYPOTHESIS: Assume that $S(k-1)$ is true.
INDUCTION STEP: Prove that $S(k)$ is true.
CONCLUSION: $S(n)$ is true for all $n \geq 0$.

(c) [2 marks]

Is the following proof structure valid? Circle one answer. You will get 2 marks for the correct answer, and -2 marks for the incorrect answer.

Yes No

Answer: No.

BASE CASE(S): Prove that $S(0)$ is true.
Let k be an arbitrary integer ≥ 0.
INDUCTION HYPOTHESIS: Assume that $S(k)$ is true.
INDUCTION STEP: Prove that $S(k*2)$ is true.
CONCLUSION: $S(n)$ is true for all $n \geq 0$.

(d) [2 marks]

Fill in the blanks to make the following proof structure valid.

BASE CASE(S): Prove that _____.
Answer: $S(0)$ and $S(1)$ are true.
Let k be an arbitrary integer $\geq$ _____.
Answer: 0.

INDUCTION HYPOTHESIS: Assume that $S(k)$ is true.
INDUCTION STEP: Prove that $S(k+2)$ is true.
CONCLUSION: $S(n)$ is true for all $n \geq 0$.

(e) [2 marks]

Fill in the blanks to make the following a valid proof using *strong* induction.

BASE CASE(S): Prove that $S(0)$ is true.
Let k be an arbitrary integer $\geq$ _____.
Answer: 0.
INDUCTION HYPOTHESIS: Assume that _____.
Answer: $S(i)$ is true for all $0 \leq i \leq k$.

INDUCTION STEP: Prove that $S(k+1)$ is true.
CONCLUSION: $S(n)$ is true for all $n \geq 0$.

3. [13 marks]

Assume the following declaration has been made:

```
class Node {
    int value;
    Node left, right;
}
```

(a) [3 marks]

Some of the operations that can be performed on a queue are:

`enqueue(o)` : add `Object o` to the end of the queue.

`dequeue()` : Remove and return the `Object` at the front of the queue.

`isEmpty()` : returns true if queue is empty, false otherwise.

Let `root` be the root `Node` of a binary tree. Which search order does the following code use?

```
Queue q = new Queue();
q.enqueue(root);
while( !q.isEmpty() ) {
    Node t = (Node)q.dequeue();
    System.out.print(t);
    if (t.left != null) q.enqueue(t.left);
    if (t.right != null) q.enqueue(t.right);
}
```

Circle the correct answer. You will receive 3 marks for the correct answer, and -2 marks if you circle the incorrect answer.

`breadth first search` `depth first search` `neither`

Answer: BFS

(b) [2 marks]

Recall that a binary tree is a tree where every node has 0, 1, or 2 children. Describe what a nonempty binary tree must look like if a preorder traversal and an inorder traversal would result in the nodes being printed in exactly the same order. No marks will be given for diagrams (although they may help you collect your thoughts); you must describe it in words.

Answer: Every node except the root is a right child. (just a right path)

(c) [2 marks]

Describe what a nonempty binary tree must look like if a postorder traversal and a preorder traversal would result in the nodes being printed in exactly the same order. No marks will be given for diagrams (although they may help you collect your thoughts); you must describe it in words.

Answer: Only one node (the root).

(d) [6 marks]

Complete the following recursive method so that it meets its specification. For full credit, only two calls to `print()` may appear inside your method. Do not use a loop or any helper methods.

```
// If 'order' is 0, print the values in the tree rooted at 'root' in preorder.
// If 'order' is 1, print the values in the tree rooted at 'root' in inorder.
// If 'order' is 2, print the values in the tree rooted at 'root' in postorder.
public static void print(Node root, int order) {
```

Answer:

```
    if (root != null) {
        if (order == 0) System.out.println(root.value);
        print(root.left, order);
        if (order == 1) System.out.println(root.value);
        print(root.right, order);
        if (order == 2) System.out.println(root.value);
    }
}
```

4. [6 marks]

Assume the following declaration has been made:

```
class Node {
    Object data;
    Node prev;
}
```

Fill in the body of method `ListWithMarker.insertBeforeMarker()` below.

```
// Maintain a list of Objects and a marker for the current position
// within that list.
class ListWithMarker {
    // The last Node in my backward singly-linked list.
    Node last;

    // A reference to a Node in my list.
    // last != null implies marker != null.
    // last == null implies marker == null.
    Node marker;

    // Insert n into my list just before Node marker.
    // Precondition: n != null
    void insertBeforeMarker( Node n ) {
```

Answer:

```
    if (marker == null) {
        last = n;
        n.prev = null;
    } else {
        n.prev = marker.prev;
        marker.prev = n;
    }
```

```
   }
}
```

5. [12 marks]

There are two instructors teaching a hypothetical course, which we will call CSC148. There are several million duties that need to be done by these instructors, and each duty has a unique id number. These duties have been divided up between the instructors, but due to a processing error one duty was placed in both lists. The duty ids for the first professor are listed in a sorted array, called `a`, and the duty ids for the second professor are listed in another sorted array, `b`. (Both arrays are in nondecreasing order.)

We can rephrase "one duty id exists in both lists" as "`for some m and n, a[m] == b[n]`". Note that we don't know what the values of `m` and `n` are — we're looking for them. But we can certainly use `m` and `n` in our assertions; they just won't appear in the code.

Your duty is to find the duplicate duty id. (Without changing the contents of the arrays.)

Follow the steps at the bottom of the page to complete the following loop.

```
// Precondition: a and b are both sorted, and for some m and n, a[m] == b[n].




// Invariant:


while (                          ) {




}

// Postcondition: a[0 .. i-1] and b[0 .. j-1] do not contain the duplicate
// value, and a[i] == b[j].
```

(a) [2 marks]

Develop the invariant from the postcondition by deleting a conjunct.

Answer:

```
Delete [i] == b[j]: a[0 .. i-1] and b[0 .. j-1] do not contain the
duplicate value.
```

(b) [2 marks]

Fill in the initialization. Answer: `i = 0;  j = 0;`

(c) [2 marks]

Fill in the loop guard. Answer: `a[i] != b[j]` (or `!(a[i] == b[j])`)

(d) [2 marks]

Write down a bound function for this loop (it should involve `m, n, i and j`):_____

Answer: `m-i  + n-j`

(e) [2 marks]

Inside the loop body, write down two statements that decrease the bound function.

Answer: `i++, j++`

(f) [2 marks]

Now add appropriate condition(s) to ensure that the invariant is maintained.

Answer:

```
if (a[i] < b[j])      i++;
else if (a[i] > b[j]) j++;
```

6: Deleted

6. [6 marks]

```
class Fnerg {
    int size = 12;
    void dig(int k) {  size = size - k;  }
    void dug(int k) {  size = size + k;  }
    public String toString() {  return "" + size;  }  // convert size to a String.
}

class Banash extends Fnerg {
    void dug(int k) {  size = size * k;  }
}

public class Snurble {
    public static void main(String[] args) {
        Fnerg tooby = new Fnerg();
        tooby.dig(4);
        System.out.println("Z: " + tooby.toString());

        Banash hibble;
        hibble = new Banash();
        hibble.dig(2);
        System.out.println("X: " + hibble.toString());

        tooby = new Banash();
        tooby.dug(10);
        System.out.println("F: " + tooby.toString());

        Banash kibble;
        System.out.println("W: " + kibble.toString());

        hibble = (Banash)tooby;
        hibble.dug(3);
        System.out.println("B: " + hibble.toString());
        System.out.println("K: " + tooby.toString());
    }
}
```

(a) [1 marks]

In the code above, cross out the only line of code that causes an error. Answer: Line `W` uses uninitialized reference.

(b) [5 marks]

Assuming that the line you crossed out in part **(a)** doesn't exist, show the output of this program:

Answer:

```
Z: 8
X: 10
F: 120
B: 360
K: 360
```

7. [14 marks]

Definitions: $0! = 1$, and for all a, $a^0 = 1$.

Consider the following method:

```
public double f(double x, int n) {
   if (n == 0) return 1.0;
   else return x * n * f(x,n-1);
}
```

(a) [2 marks]

Consider a call `f(x,n)`. For what values of `n` does `f(x,n)` terminate? Answer: $n \geq 0$

(b) [2 marks]

Give a precise description of what `f(x,n)` computes. (**Hint:** A factorial is part of the answer.) Answer: $n! * x^n$

(c) [10 marks]

Prove that `f(x,n)` terminates for those values of `n` that you gave in **(a)**, and that it returns what you claim in **(b)**. Note that you can get part marks for proving termination even if you didn't get the answer to **(b)**. Be sure to give a precise description of what you are trying to prove.

Answer:

The proof is by induction on n.

Let $S(n)$ be the statement: For every integer $n \geq 0$, and every real number x, `f(x,n)` terminates, and returns $n! * x^n$.

Base Case: $n = 0$. Then `f(x,0)` terminates, and returns 1.0, which is $0! * x^0$ for every real number x. Therefore, $S(0)$ is true.

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

Induction Step: We prove that $S(k+1)$ is true.

Let x be any real number, and consider the call `f(x,k + 1)`. Now $k \geq 0$; therefore, $k+1 \neq 0$, so the call `f(x,k + 1)` returns $x * (k+1) *$ the value of the call `f(x,k)`. By the induction hypothesis, the call `f(x,k)` terminates, and returns $k! * x^k$. Therefore, the call `f(x,k + 1)` terminates, and returns $x * (k+1) * k! * x^k = (k+1)! * x^{k+1}$. Therefore, $S(k+1)$ is true, and the proof is complete.

8. [6 marks]

Here are the first five rows of Pascal's triangle:

```
row 0:     1
row 1:     1   1
row 2:     1   2   1
row 3:     1   3   3   1
row 4:     1   4   6   4   1
row 5:     1   5   10  10  5   1
```

The numbers on the ends of rows are all 1. Each other number is the sum of the number directly above it and the number above and to the left. We can index into the triangle like this:

Table 1:

```
      P(0,0)


      P(1,0)    P(1,1)
        3         3

      P(2,0)    P(2,1)    P(2,2)
        2         3         1

      P(3,0)    P(3,1)    P(3,2)    P(3,3)
        1         2         1

      P(4,0)    P(4,1)    P(4,2)    P(4,3)    P(4,4)
                  1         1

      P(5,0)    P(5,1)    P(5,2)    P(5,3)    P(5,4)    P(5,5)
                            1
```

For example, `P(4,2)` is 6.

In general, row i looks like this:

$$\texttt{P}(i,0) \qquad \texttt{P}(i,1) \qquad \cdots \qquad \texttt{P}(i,i)$$

Consider the following method.

```
// Returns P(i,j).
// Precondition: 0 <= j <= i.
public static int P(int i, int j) {
    if ((j == 0) || (j == i)) {
        return 1;
    } else {
        return P(i-1, j-1) + P(i-1, j);
    }
}
```

Consider a call to `P(5,2)`. In Table 1 above, for each `i` and `j`, indicate the number of times `P(i,j)` is evaluated, by writing the number of evaluations underneath `P(i,j)`. To get you started, `P(5,2)` is calculated 1 time, so you should write `1` directly underneath `P(5,2)`.

9. [8 marks]

A deque is an abstract data type that works a bit like a stack or queue, except that insertions and deletions are allowed at *both* ends. The operations that can be performed on a deque are:

`size()`: return the number of objects in the deque.
`insertAtFront(o)`: given an `Object o`, put it on the front end of the deque.
`insertAtBack(o)`: given an `Object o`, put it on the back end of the deque.
`removeFront()`: remove and return the element at the front of the deque.
`removeBack()`: remove and return the element at the back of the deque.

(a) [2 marks]

Consider the following code fragment:

```
Deque d = new Deque();
d.insertAtFront(new Integer(2));
Integer i = (Integer)d.removeBack();
d.insertAtBack(new Integer(3));
d.insertAtBack(new Integer(4));
d.insertAtFront(new Integer(1));
i = (Integer)d.removeBack();
```

Show the resulting contents of `d`. Be sure to indicate which end is which.

`Contents of d:`

Answer: `Front: 1  Back: 3` (And that's it.)

(b) [3 marks]

Recall that `i % j` is the remainder after dividing `i` by `j` — the answer can be negative if `i` is negative.

Assuming that the following code for `class Deque` has been written, complete the method `insertAtFront()`. You may not use any form of an `if`-statement. Do not worry about an overfull array.

```
public class Deque {
    private final int MAX_SIZE = 10;
    private int front = 1;
    private int back = 0;

    // 'contents' is a circular array.
    // contents[front .. back] hold the elements in me.
    private Object[] contents = new Object[MAX_SIZE];

    public void insertAtBack(Object o) {
        back = (back+1) % MAX_SIZE;
        contents[back] = o;
    }

    public void insertAtFront(Object o) {
```

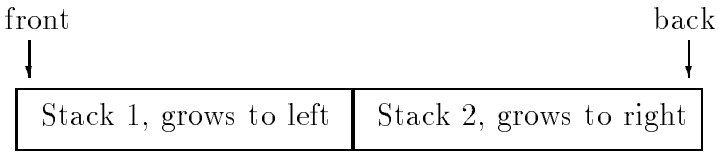
Answer:

```
        front = (front-1 + MAX_SIZE) % MAX_SIZE;
        contents[front] = o;


    }
```

(c) [3 marks]

Assume that `class Stack` has methods `push()` and `pop()` which each have O(1) running time. Can a deque be implemented using two `Stack`s as instance variables, one for each end of the deque, such that all insertions and removals have O(1) running time?



A fast deque implementation?

Circle the correct answer; note that you will receive no marks unless your explanation is clear:

`yes` `no`

Briefly explain your answer:

Answer: No, because if one `Stack` is empty then the next access would be to the bottom of the other `Stack`, which is not O(1).

We hope that you had a pleasant term, and wish you a great summer!

Total Marks = 87