UNIVERSITY OF TORONTO
Faculty of Arts and Science

December Examinations 2000

CSC148F: Introduction to Computer Science

Duration: 3 hours

Aids allowed: None.

- **Make sure your examination booklet has 13 pages (not including this one).**
- Write your answers in the spaces provided. Do not feel that you must use all of the space provided. If you run out of space on any question, use the back of a page, and draw an arrow to point this out.
- You will be rewarded for answers that are concise and well thought out, rather than long and rambling.
- Write legibly. Unreadable answers will be given 0.
- Unless explicitly asked for, comments are not necessary.

Family Name: ____________________ Given Names: ____________________

Student #: ____________________ Lecturer: ____________________

TA:						
	Morning:	A–C Listgarten	D–Kl Godil	Km–Me Imrisek	Mf–Si Sadikali	Sj–Z Dalmao
	Afternoon:	A–Hua Bolintineanu	Hub–Pen Oakham	Peo–Z Wigdor		

1. ________ / 14
2. ________ / 11
3. ________ / 9
4. ________ / 14
5. ________ / 8
6. ________ / 7
7. ________ / 8
8. ________ / 8
9. ________ / 15
10. ________ / 6

Total ________ / 100

Good Luck!

Question 1

[14 marks in total]

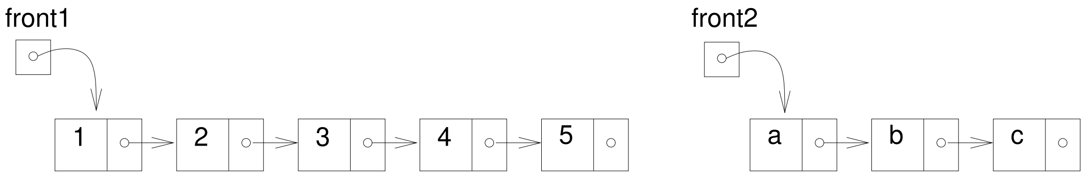
Suppose we have this class for nodes in a linked list:

```
class ListNode {
    Object data;
    ListNode link;
}
```

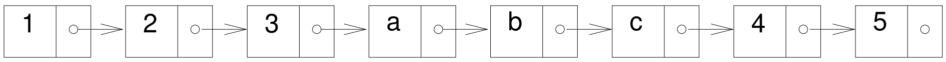
Here is the heading for a method that will “include” one linked list into the other, after a given position n.

```
public static ListNode include(ListNode front1, int n, ListNode front2) {
```

For example, if the two lists are as follows:



and n is 3, the method should return a reference to the front node of this list:



Note that this method just moves nodes around. It **does not create any new nodes**, or make new copies of any node’s contents.

(a) [4 marks]
Read the full specifications for the method, given in the comments for part (b), and then describe a good testing strategy for this method.

(b) [10 marks]
Write the method body.

```
// Let list1 be the list whose first node is front1, and
// let list2 be the list whose first node is front2.

// Include list2 into list1, after the nth node of list1, and return a
// reference to the front node of the resulting list.
// If n is 0, include list2 at the very front of list1.

// Preconditions: 0 <= n <= the number of nodes in list1, and
//                list2 has at least 1 node.  (But list1 may be empty.)

public static ListNode include(ListNode front1, int n, ListNode front2) {
```

Question 2

[11 marks in total]

(a) [2 marks each]
What is the worst-case big-oh time complexity of each of the following problems? Assume that the problems are solved efficiently using appropriate algorithms.

Given an integer array of length n, print the array in reverse order.

ANSWER: ________________________

Given a queue containing q items (and represented as a circular array), and a stack containing s items (and represented as a linked list), remove the head of the queue and push it onto the stack.

ANSWER: ________________________

Given a tree of n nodes and height h, return the number of occurrences of a given object.

ANSWER: ________________________

Given a binary search tree of n nodes and height h, return the maximum value in the tree.

ANSWER: ________________________

(b) [3 marks]
Suppose that we have declared all the variables that appear in the code below, and that we have read values into the integers r and m. Suppose also that the comments in the following code are replaced with appropriate method calls:

```
p = 15;
for (int i = 0; i < r; i++) {
    k = m;
    while (k > 0) {
        if (p % 4 == 3) {
            // Insert p to the beginning of a linked list L,
            // containing n nodes.
        }
        else {
            // Search for p in a linked list L,
            // containing n nodes.
            p = p++;
        }
        k--;
    }
}
```

What is the worst-case big-oh time complexity of this code? Give the strongest answer you can, that is, the tightest bound.

ANSWER: ________________________

Question 3

[9 marks in total]

(a) [2 marks]
Circle your TA's name on the front page, and write your name and student number legibly at the top of *every* page of this exam.

(b) [3 marks]
Which of the following statements are true? Circle TRUE or FALSE for each.

A binary tree must have exactly 1 root	TRUE	FALSE
Every node in a binary tree must have exactly 1 parent	TRUE	FALSE
Every node in a binary tree must have exactly 2 children	TRUE	FALSE

(c) [4 marks]
Suppose we were to build a trie, as in your project, but with this alphabet of 16 characters:

$$0123456789() + - * /$$

And suppose we picked a context size (also called k) of 4. What is the greatest number of leaves that the trie might have? You need not simplify your answer; a formula is fine.

Answer: ______________

What is the greatest number of nodes in total that the trie might have? You need not simplify your answer; a formula is fine.

Answer: ______________

Question 4

[14 marks in total]

A "rude queue" is like a regular queue except that it supports these two additional operations:

- sneakIn(obj): Allow obj to "sneak in" to front of the queue. That is, put obj at the front of the queue, ahead of all the other objects. An exception occurs if the queue is full.
- goBack(): Move the object at the head of the queue to the tail. An exception occurs if the queue is empty.

On page 6 is a `Queue` interface and a `CircularQueue` class. You are going to build on this code.

(a) [5 marks]
Define a new interface called `RudeQueue` that extends `Queue` and supports the additional operations described above.

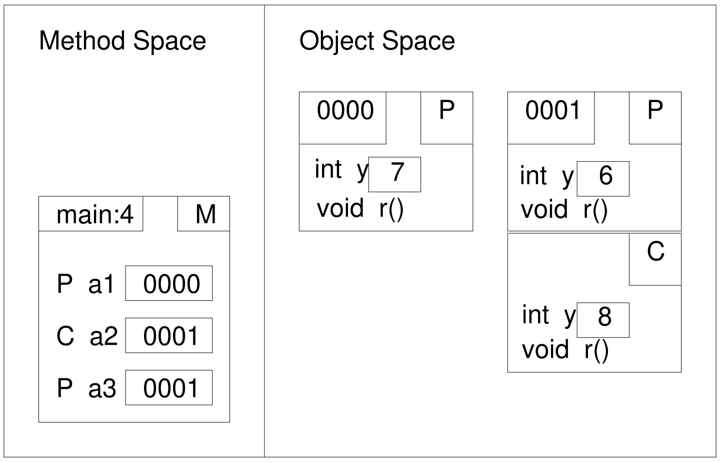
(b) [9 marks]
Write a new class called `CircularRudeQueue` that extends the `CircularQueue` class and implements your `RudeQueue` interface. For full credit, you must make use of as much of the parent class `CircularQueue` as possible. Note that `Queue.enqueue()` throws a `QueueFullException`. If you use this method, you must catch the exception; just print a message if it occurs.

```
public class QueueFullException extends Exception {}
public class QueueEmptyException extends Exception {}

public interface Queue {                          // adapted from lecture notes.
    // Append o to me.
    public void enqueue(Object o) throws QueueFullException;

    // Remove and return my front Object.
    public Object dequeue() throws QueueEmptyException;

    // Return the number of Objects in me.
    public int size();
}

public class CircularQueue implements Queue {     // adapted from lecture notes.
    // The maximum number of elements I can hold.
    protected static final int Capacity = 100;
    // The number of elements in me.
    protected int size = 0;
    // The index of the head and tail of the queue.
    protected int head, tail = 0;
    // The items in me.
    protected Object[] contents = new Object[Capacity];

    // Representation invariant:
    // If size is 0, I am empty and head = tail.
    // Otherwise:
    //    contents[head] is the head.
    //    contents[(tail+Capacity-1) % Capacity] is the tail.
    //    if head < tail,
    //       contents[head .. tail-1] contains the Objects in the order they
    //       were inserted, and size = tail - head.
    //    if head >= tail,
    //       contents[head .. Capacity-1, 0 .. tail-1] contains the Objects
    //       in the order they were inserted, and size = tail - head + Capacity.

    public void enqueue(Object o) throws QueueFullException {
        if (size==Capacity) throw new QueueFullException();
        contents[tail] = o;
        tail = (tail+1) % Capacity;
        size++;
    }

    public Object dequeue() throws QueueEmptyException {
        if (size==0) throw new QueueEmptyException();
        Object result = contents[head];
        head = (head+1) % Capacity;
        size--;
        return result;
    }

    public int size() { return size; }
}
```

Question 5

[8 marks in total; 2 marks each]

Suppose we are executing a Java program, and at this moment the memory model looks as shown below. (The static space is not included because it is not relevant.)



Circle the best answer to each of the following multiple-choice questions.

To access `P`'s `y` using `a2`, I need to do
(1) `a2.y` (2) `a2.(P)y` (3) `((P)a2).y` (4) `a2.super.y`
(5) none of the above works, but it can be done
(6) it cannot be done

To access `C`'s `y` using `a3`, I need to do
(1) `a3.y` (2) `a3.this.y` (3) `a3.(C)y` (4) `((C)a3).y`
(5) none of the above, but it can be done
(6) it cannot be done

To call `P`'s `r()` using `a2`, I need to do
(1) `a2.r()` (2) `a2.(P)r()` (3) `((P)a2).r()` (4) `a2.super.r()`
(5) none of the above works, but it can be done
(6) it cannot be done

To call `C`'s `r()` using `a3`, I need to do
(1) `a3.r()` (2) `a3.this.r()` (3) `a3.(C)r()` (4) `((C)a3).r()`
(5) none of the above works, but it can be done
(6) it cannot be done

Question 6

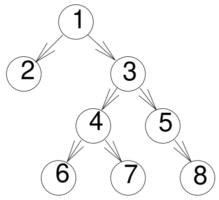
[7 marks in total]

Consider the following method, and the BinaryIntNode class it uses.

```
public static int mystery(BinaryIntNode root) {
    if (root.left == null && root.right == null)
        return root.key;
    else if (root.left == null)
        return mystery(root.right);
    else if (root.right == null)
        return mystery(root.left);
    else
        return mystery(root.left) + mystery(root.right);
}
```

```
class BinaryIntNode {
    int key;
    BinaryIntNode left;
    BinaryIntNode right;
}
```

(a) [2 marks]
What value is returned if we call mystery with a reference to the root of this binary tree:



ANSWER: ____________________

(b) [2 marks]
How many calls to mystery occur, in total, as a result of that initial call to mystery? Include the initial call in your total.

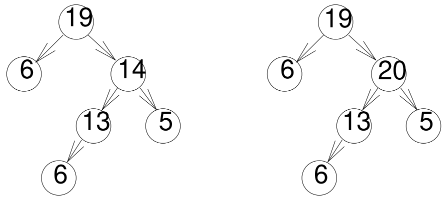
ANSWER: ____________________

(c) [3 marks]
Write an appropriate external comment for mystery.

Question 7

[8 marks in total]

A binary tree is "top heavy" iff, for every node in the tree, that node has a key greater than its children do (to the extent that it has children). For example, the tree on the left is top heavy and the tree on the right is not:



An empty tree is considered to be top heavy.

Complete the following method. It uses the `BinaryIntNode` class from page 8.

Hint: Use recursion.

```
// Returns whether or not the binary tree whose root is ‘root’ is top heavy.
public static boolean isTopHeavy(BinaryIntNode root) {
```

Question 8

[8 marks in total]

(a) [4 marks]
Consider the following outline of a proof:

Assume A is true
    ...
    Therefore B is true
Assume A is false and B is false
    ...
    Therefore some contradiction holds

If the missing parts (marked "...") are filled in correctly, what does the first half prove?

ANSWER: ______________________________

If the missing parts (marked "...") are filled in correctly, what does the second half prove?

ANSWER: ______________________________

(b) [2 marks]
From the two conclusions above, what conclusion can be proven? Circle one.

1. $(B \Rightarrow A)$ is true
2. $(B \Rightarrow A)$ is false
3. B is true
4. B is false
5. None of the above

(c) [2 marks]
Consider this proof structure:

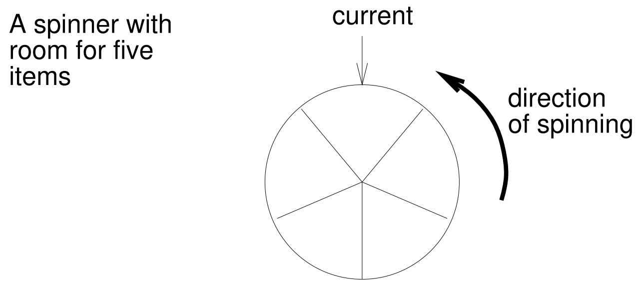
Base Case:	Prove that $S(3)$ is true.
Let k be an arbitrary integer > 3.	
Induction Hypothesis:	Assume that $S(k)$ is true.
Induction Step:	Prove that $S(k+2)$ is also true.

Assuming the details of the base case and induction step sub-proofs are filled in correctly, which of the following conclusions would be justified by the proof? Circle each one that is justified.

$S(3)$ $\quad$ $S(4)$ $\quad$ $S(5)$ $\quad$ $S(6)$ $\quad$ $S(7)$ $\quad$ $S(8)$

Question 9
[15 marks in total]

Consider the following new ADT: A **spinner** is a circular container that holds a fixed number of objects, and has a "current" position, which is initially 0. You can picture a spinner like this:



The following operations can be performed on a spinner:

- stash(o): Replace the object at the current position (if any) with o.
- look(): Return the item at the current position.
- spin(n): Spin ahead n positions.
- unstash(): Undo the last stash operation. That is, make the current position be what it was before the last stash operation, and restore the contents there to what it was.

Here is an interface for spinner:

```
interface Spinner {
    // Replace the object at my current position (if any) with o.
    public void stash (Object o);

    // Return the item at my current position, or null if nothing is there.
    public Object look ();

    // Spin ahead n positions.
    // Precondition: n >= 0.
    public void spin(int n);

    // Make my current position be what it was before the last stash operation, and
    // restore the contents there to what it was.
    // This method may only be called *once* after a stash.
    // Precondition: This is the first call to unstash since the last call to stash.
    public void unstash();
}
```

On the next page, write a class that implements `Spinner` using a circular array. Write one constructor. It should take as an argument the desired capacity of the spinner.

Comments are not necessary on your methods, but you must comment your instance variables and write a representation invariant.

Answer for Question 9.

Question 10

[6 marks in total]

Circle the one best answer for each of questions below. They concern the following outline of code:

```
interface Blah { ... }
class B implements Blah { ... }
abstract class Clem { ... }
class C extends Clem
class Dreft { ... }
class D extends Dreft { ... }
```

(a) Must `B` implement all unimplemented methods of `Blah`?

1. Yes.
2. No. If it doesn't, we must not make an instance of `B`.
3. No. If it doesn't, we can make an instance of `B`, but we must not call the unimplemented methods.
4. `Blah` cannot have any unimplemented methods.

(b) Can `Blah` declare any variables?

1. No, none of any sort.
2. It may declare static variables but not instance variables.
3. It may declare instance variables but not static variables.
4. It may declare both static variables and instance variables.

(c) Can `Clem` declare any variables?

1. No, none of any sort.
2. It may declare static variables but not instance variables.
3. It may declare instance variables but not static variables.
4. It may declare both static variables and instance variables.

(d) Must `D` implement all unimplemented methods of `Dreft`?

1. Yes.
2. No. If it doesn't, we must not make an instance of `D`.
3. No. If it doesn't, we can make an instance of `D`, but we must not call the unimplemented methods.
4. `Dreft` cannot have any unimplemented methods.

(e) Can I write a class that extends both `Dreft` and `Clem`? YES NO

(f) Can I write a class that both implements `Blah` and extends `Dreft`? YES NO

END OF EXAM; Total pages = (13)