PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER 2002 EXAMINATIONS

CSC 148H1 F / 150H1 F

Duration — 3 hours

Examination Aids: One 8.5" × 11" sheet of paper, *hand*written on both sides.

Student Number: |_|_|_|_|_|_|_|_|_|_|

Last Name: _____

First Name: _____

Course enrolment: **(circle one)** CSC 148H1 F CSC 150H1 F

---

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

This final examination consists of 8 questions on 13 pages (including this one), printed on one side of the paper. *When you receive the signal to start, please make sure that your copy of the examination is complete.*

Answer each question directly on the examination 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 the part of your work that should be marked.*

Unless otherwise specified, you are allowed to write helper methods and comments are *not* required. However, comments can be helpful when assigning part marks.

If you are unable to answer a question, or you can only give a partial answer, you may get up to 20% of the marks for the question if you state clearly that you do not know how to answer. Note that you will *not* get those marks if your answer contains contradictory statements (such as "I do not know how to answer" followed or preceded by a solution).

General Hint: We were careful to leave ample space on the examination paper to answer each question. Unless we explicitly ask, you do not have to write comments for your code. Also, remember that hints are just hints: you are not required to follow them if you can think of a different solution, and they are generally only *part* of a complete answer.

# 1:	_____/ 16
# 2:	_____/ 16
# 3:	_____/ 14
# 4:	_____/ 9
# 5:	_____/ 8
# 6:	_____/ 10
# 7:	_____/ 20
# 8:	_____/ 12
BONUS:	_____/ 5
TOTAL:	_____/105

Good Luck!

Question 1. [16 MARKS]

Part (a) [3 MARKS]

Complete the identification section at the top of page 1, then write your student number **legibly** at the bottom of every page of this examination except page 1 (where indicated).
(HINT: The questions on this examination are in no particular order; start with the easier questions first!)

Part (b) [13 MARKS]

Circle the correct answer for each of the following questions. (Note that according to our "20% rule", you will get 0.2 marks for each answer left blank, compared with 0 for each incorrect answer.)

TRUE **FALSE** A Java interface can extend another Java interface.

TRUE **FALSE** A queue is a sorted abstract data type.

TRUE **FALSE** A linked list is the best implementation of a queue ADT because it can be searched using binary search.

TRUE **FALSE** The Java `Iterator` interface has three methods: `hasNext`, `next`, and `remove`.

TRUE **FALSE** External comments should describe implementation details, including a specific data structure description.

TRUE **FALSE** Every `throw` statement must have a corresponding `catch` statement.

TRUE **FALSE** Method overriding is the same as method overloading.

TRUE **FALSE** A Java interface is the same as an abstract data type.

TRUE **FALSE** `Comparable` is an interface that is part of the Java API.

TRUE **FALSE** A map of roads and cities can be represented using a graph ADT.

TRUE **FALSE** Nested loops are always $\mathcal{O}(n^2)$.

TRUE **FALSE** An algorithm that is $\mathcal{O}(n)$ is also $\mathcal{O}(n^2)$.

TRUE **FALSE** Searching a linked list is $\mathcal{O}(\log_2(m))$ where m is the length of the list.

Question 2. [16 MARKS]

Consider the following `IntNode` class, which can be used to implement linked lists of integers.

```
public class IntNode {
    public int data;
    public IntNode next;
    public IntNode(int i)  { data = i; }
}
```

Part (a) [8 MARKS]

Complete the following method according to its specification. (You should not need all of the space below for your answer.)

```
/* Return a node that contains the smallest integer in the list referred to by 'first'.
   Return null if the list referred to by 'first' is empty.
*/
public static IntNode findMin(IntNode first) {




} // end findMin()
```

Question 2. (CONTINUED)

Part (b) [8 MARKS]

Complete the following method according to its specification. You may make use of the `findMin()` method from part (a) to do this. However, you do not need to complete part (a) in order to answer this part. (You should not need all of the space below for your answer.)

```
/* Sort the list of integers referred to by 'first', in non-decreasing order.
   Return a reference to the front of the sorted list.
*/
public static IntNode selectionSort(IntNode first) {
    // HINT: You can directly change the data field of nodes.
```

```
} // end selectionSort()
```

Question 3. [14 MARKS]

In the space below, write a static Java method called `getInt` that reads input from a `BufferedReader` passed in as argument until a valid integer is read, then returns that integer. If the user enters something that is not a valid integer, you should print a message asking the user to try again, and read another line from the `BufferedReader`. If the end of input is reached before the user enters a valid integer, your method should throw a `NullPointerException` (a predefined subclass of `RuntimeException`). You do not need to handle I/O errors that may occur, and you may assume that your method belongs to a class where appropriate `import` statements have already been made.

In your answer, you may call the following two methods as appropriate:

- `static int parseInt(String s)`
  A method from the `Integer` class that returns the `int` represented by `s`. If `s` does not represent a valid integer, it throws `NumberFormatException`.
- `String readLine()`
  A method from the `BufferedReader` class that fetches characters from the input until a newline is read. It returns the characters read—not including the newline—or `null` if the end of the input has already been reached. It throws `IOException` if an I/O error occurs.

Question 4. [9 MARKS]

Consider the following Java program.

```
class Some {

    private static int count = 0;
    private Some someClass = null;
    private int number;

    public Some(int n) {
        count++;                                  // line 1
        number = n;                               // line 2
        if (number > 1)                           // line 3
            someClass = new Some(number - 1);     // line 4
    }

    public int someIt() {
        if (someClass != null)                    // line 1
            return number + someClass.someIt();   // line 2
        else
            return number;                        // line 3  [See Part (b)]
    }

    public static int otherSome() {
        return (count * (count + 1)) / 2;         // line 1
    }

}

class TestSome {
    public static void main(String[] args) {
        Some someClass = new Some(3);                 // line 1
        if (someClass.someIt() == Some.otherSome())   // line 2
            System.out.println("Is equal");           // line 3
        else
            System.out.println("Not equal");          // line 4
    }
}
```

Part (a) [1 MARK]

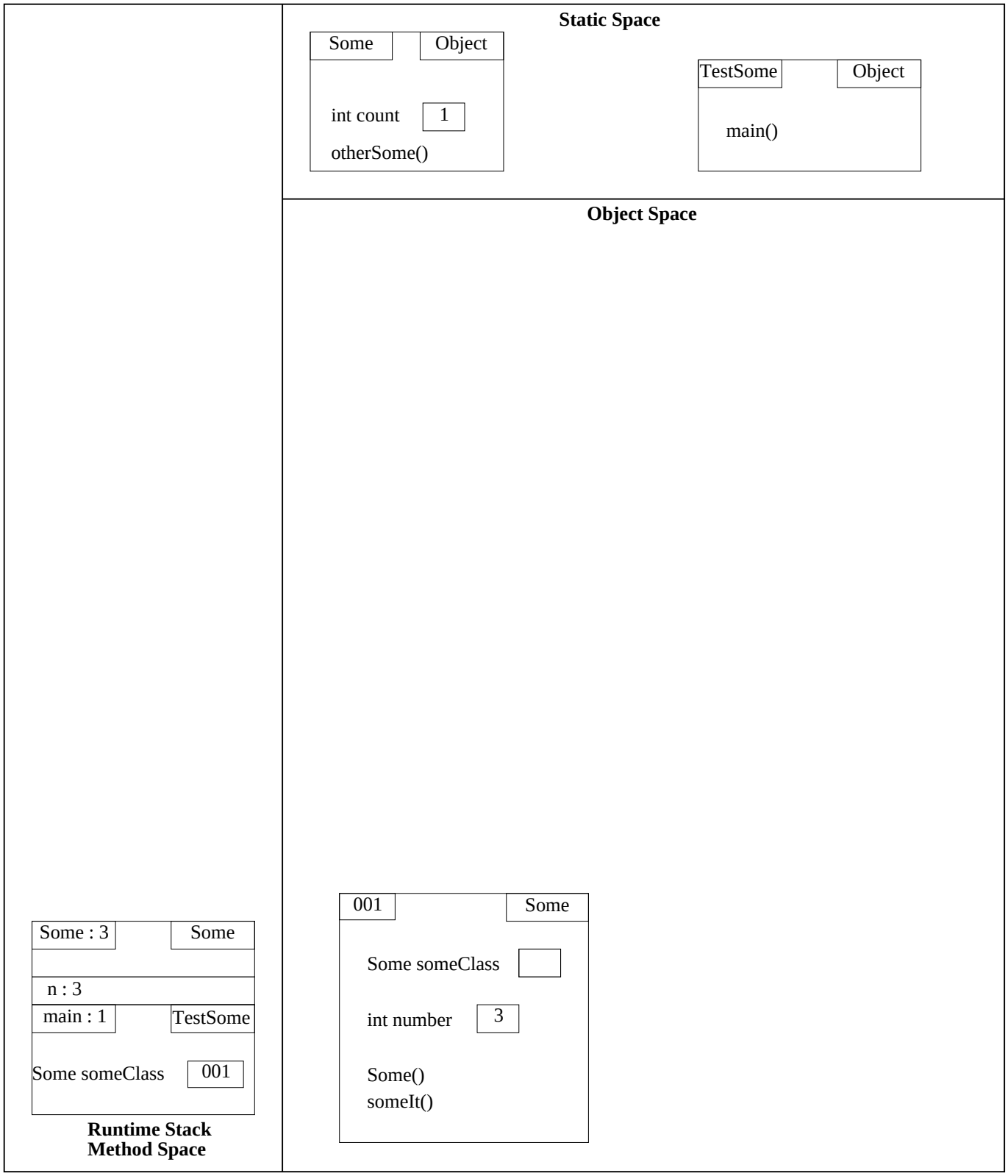
Write the output of the program when it is executed to completion.
(HINT: You may find it helpful to do part (b) first.)

Part (b) [8 MARKS]

On the next page, complete the memory model trace of the program above. Illustrate the state of the memory model when execution reaches the line labelled `[See Part (b)]` (*i.e.*, just *before* that line is executed). On your picture, cross out anything that has changed or disappeared.

Question 4. (CONTINUED)

Part (b) (CONTINUED)



Question 5. [8 MARKS]

Recall that a binary tree is called a "heap" if for every node in the tree, the label at the node is less than or equal to the labels of its children (if they exist)—such a binary tree is said to satisfy the "heap property". For this question, we use the following `BinTreeNode` class for the nodes in a binary tree.

```
class BinTreeNode {
    Comparable data;
    BinTreeNode left, right;
}
```

Write the body of the following method according to its specification. (You should not need all of the space below for your answer.)

```
/* Return true if the binary tree rooted at 'root' is a heap; false otherwise.
   Requires: root != null
*/
public static boolean isHeap(BinTreeNode root) {
```

```
} // end isHeap()
```

Question 6. [10 MARKS]

Suppose you were considering purchasing a class called `WizBangPQ` which implements the `PriorityQueue` interface from Assignment 4 (reproduced below for your reference).

```
/** A collection of elements ordered by "priority".  The relative priority
    of elements is determined with the 'compareTo' method.  */
public interface PriorityQueue {

    /** Return true if I am empty; false otherwise. */
    boolean isEmpty();

    /** Insert 'item' in my elements, in the proper order. */
    void insert(Comparable item);

    /** Remove and return my smallest element.  Requires: I am not empty.
        @throws  NoSuchElementException  if I am empty  */
    Comparable deleteMin();

}
```

Design test cases to verify the correctness of `WizBangPQ`. Briefly explain the significance of each of your test cases (what does it test, and why is it important).
(HINT: Don't spend too long on this question! You only need a small number of well-chosen cases to get full marks.)

Question 7. [20 MARKS]

Consider the following description of a real-world system that you must represent in a Java program.

- A train has a sequence of cars where each car has a length. A train has passenger cars and freight cars. A passenger car can carry at most 75 passengers, and the weight of a passenger car is determined by the number of passengers it carries. The weight of a freight car is determined by its load.
- We can add a car to a train, add passengers to a car, add freight to a car, find out the length of the train, find out how many passengers are on the train, find out how much freight the train is carrying, and find the total weight of the train.

Part (a) [3 MARKS]

Which of the following data structures or ADTs would be a good choice to hold the sequence of cars: array, vector, linked list, queue, stack, tree? Explain your choice. (There is more than one correct answer; you only need one.)

Part (b) [2 MARKS]

In the class design you will complete on the next page, the class `Car` is declared as an abstract class. Describe two differences between an abstract class and a regular class.

Part (c) [2 MARKS]

Suppose we want to write a method `heaviest` that returns a reference to the heaviest car in the train. Which class or classes would need a `compareTo` method, to allow us to implement method `heaviest`?

Part (d) [1 MARK]

What is the most appropriate class in which to put method `heaviest`?

Question 7. (CONTINUED)

Part (e) [12 MARKS]

Below, design a set of classes that could be used to implement the operations described on the previous page. In particular, complete the `Car` and `Train` classes, and write the remaining classes with their instance and static variables and method *signatures*. Do *not* write the bodies of the methods, but indicate a missing body with `{...}`, as demonstrated below.

```
public abstract class Car {
    private double length;

    public double getLength()  {...}




}

public class Train {




    public double getLength()  {...}
    public double getWeight()  {...}
}
```

Question 8. [12 MARKS]

Recall the definition of the Fibonacci sequence of numbers $F_0, F_1, F_2, \ldots$:

$$F_0 = 0, \qquad F_1 = 1, \qquad F_n = F_{n-1} + F_{n-2} \quad \text{for } n \geq 2.$$

Prove that for all integers $n \geq 0$, $\sum_{i=0}^{n} F_i^2 = F_n \times F_{n+1}$. Your proof will be marked on its structure as well as its content.

Bonus. [5 MARKS]

WARNING! This question is difficult and will be marked very harshly. Part marks will be awarded only for substantial efforts. Do not try this question unless you have completed the rest of the examination.

Recall the `BinTreeNode` class from Question 5.

```
class BinTreeNode {
    Comparable data;
    BinTreeNode left, right;
}
```

Write the body of the following method according to its specification.

```
/* Rearrange the labels of the binary tree rooted at root so that the new tree
   satisfies the heap property.  The shape of the tree is not altered.
*/
public static void heapify(BinTreeNode root) {




} // end heapify()
```

Total Marks = 105