THE UNIVERSITY OF MANITOBA

Winter 2010 **Sample Final Examination**
DEPARTMENT & COURSE: COMP 1020 **PAGE :** 1 of 12
EXAMINATION: INTRODUCTORY COMPUTER SCIENCE 2 **TIME:** 2 **HOURS**

Instructions:

1. Answer all questions.
2. Any reference to "printing" a value means to use System.out.
3. Aids such as electronic calculators, PDA's, cell phones, notes, books, etc. are not permitted.
4. **Place all answers on this examination paper.**
5. There are 12 questions worth a total of **80** marks.
6. There are two parts to the exam. Part A (fill-in-the-blanks) is worth **5** marks, Part B (short answer) is worth **15** marks, Part B (programming) is worth **60** marks.
7. The last sheet in the exam is for rough work; you may separate it. Do not remove any other pages, and hand in all work you want to receive credit for!

Name	
Student Number	

Question	Mark
1	/5
2 – 3	/3
4 – 7	/12
8	/12
9	/12
10	/12
11	/12
12	/12
Exam Total: /80	

Part A: Fill-in-the-Blank

[5] 1. Fill in the blank with the most appropriate answer selected from the word list below (1 mark each):

(a) The _____ sort uses divide-and-conquer to achieve better performance than the simple sorting algorithms.

(b) A(n) _____ is a special instance method whose name is the same as the class name, and is used to initialize the contents of a newly-created object.

(c) The value of a _____ variable is shared over all instances of a class.

(d) We have two classes: Fruit and Grape. If Grape extends Fruit, we say that Fruit is a(n)

_____ of Grape.

(e) In Java, a(n) _____ class like Integer is required to store primitives in an ArrayList.

Word list: binary, bubble, constructor, encapsulation, insertion, instance, iteration, merge, null, Object, operator, recursion, refactoring, static, subclass, superclass, toString, wrapper.

Part B: Short Answer

[1] 2. Given the following simple Java class:

```
class Student {
   private int number;
   public Student(int number) {
      this.number = number;
   }
}
```

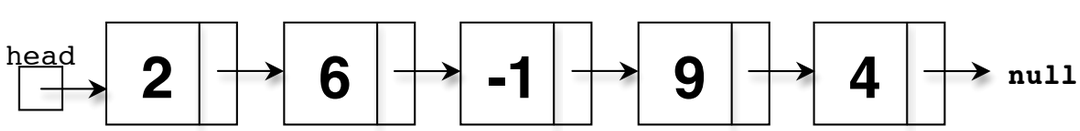
In this line of Java code:

```
Student[] array = new Student[7];
```

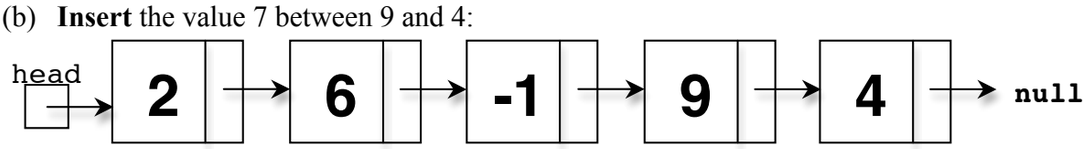
how many objects of class Student have you created? _____

[2] 3. **Alter** the diagrams of linked lists below to illustrate the changes made by the operation described. **Clearly draw** any new nodes, **cross out** any links that are broken, and **draw** any new links that are made.

(a) **Remove** the item -1:



(b) **Insert** the value 7 between 9 and 4:



---

[3] 4. Given the following list of integers that is being sorted into ascending order:

```
8 3 14 2 7
```

Show the complete contents of the array after the **first complete pass** of an in-place:

(a) **bubble sort** (left to right)

(b) **insertion sort** (assuming that the leftmost element is already in the "sorted" part)

(c) **selection sort** (the "sorted" part will start at the left)

[5] 5. (a) Arrange the following algorithm complexity measurements in order, from the **least** to the **most efficient**: O(n), O(1), $O(n^2)$

_____ _____ _____
least efficient most efficient

(b) Name two different **search** algorithms that have different efficiencies (different big-O). Give the **big-O** of each of the algorithms.

(c) Even though the insertion sort is often the fastest of the three simple **sorting algorithms** in the previous question, they all have the same big-O. Why?

[2] 6. Briefly **describe one** characteristic of an ADT.

[2] 7. Briefly **describe one plausible reason** for getting a Stack Overflow exception when you are running a Java program.

Part B: Programming

- Answer all questions using programs written in the Java language.
- Comments are not necessary, though they may help when your exam is marked.
- Use good programming techniques (make instance variables private or protected, use inheritance and/or methods to avoid duplicated code).
- Write only the code you are asked to write; do not write main programs or import statements.

[12] 8. Write a method process2DArray that accepts an *N*x*M* array filled with integer (int) values as a parameter. The method is to **return** an (*N*+1) x (*M* + 1) array which contains the contents of the original array in the first *N* rows and *M* columns plus a **count** of items **greater than or equal to zero** in each row/column at the end of that row/column. Put the value -1 in the bottom-right corner. For example:

```
1 -2  0                          1 -2  0  2
3 -4 -5          returns         3 -4 -5  1
                                 2  0  1 -1
```

---

[12] 9. Write a method that accepts an array of integers as a parameter and counts and returns the length of the longest sequence of repeated values using **recursion**. For example, the array `{ 1,2,3,3,4,2,2,2,5 }` would return 3 because of the sequence 2,2,2 (length 3).

If you need, you may write a driver method with a single array parameter and have it call your "real" recursive method with more than one parameter.

Use recursion only; no loops are allowed!

[12] 10. (a) Write a **class** called `Athlete` that keeps track of how long an athlete has taken to finish a race. It has **three instance variables**: the athlete's name (String), jersey number (int), and a finish time (double). It should also have a **constructor** to initialize all the instance variables, and a **`double`** `getTime()` method to return the time instance variable. No other methods are necessary.

Question 10, cont'd.

(b) Using the class you wrote in part (a), complete the following static method. It is passed four parameters: an ArrayList of `Athlete`s, and the name, jersey number, and finish time of another athlete. It should **create** a new `Athlete` with the given name, number, and time, and then add the new athlete to the ArrayList using an **ordered insertion**, in increasing order of finish time. Assume that the athletes already in the ArrayList are ordered by their finish time.

```
public static void addAthlete(ArrayList athletes,
                      String name, int number, double time) {
```

[12] 11. Given the following linked list `Node` class:

```
public class Node {
   private double data;
   private Node next;
   public Node(double data, Node next) {
      this.data = data;
      this.next = next;
   }
   public double getData() {
      return data;
   }
   public void setData(double data) {
      this.data = data;
   }
   public Node getNext () {
      return next;
   }
   public void setNext(Node next) {
      this.next = next;
   }
}
```

Write a `LinkedList` class that will implement a linked list of doubles. Your linked list class should have an appropriate **constructor**, and the following :

- `public void addFromArray(double[] array)` – **add all** of the values from the array to the linked list, in the same order as in the array, to the beginning of the list. It must add the strings to the list; do not overwrite any items already in the list.
- `public int count(double min, double max)` – **count** and return the number of values in the list in the in the range [`min`..`max`] (inclusive).

Question 11, cont'd

[12] 12. Implement two classes representing different types of vehicles: `Motorcycle` and `Car`, along with an appropriate superclass, as follows.

Both of these vehicles have a price (double), a model year (integer), and a gas mileage in litres per 100km (double). Cars have a number of doors (integer). These values will be passed to the constructor.

Both classes also have the following method:

- **`void`** `increaseYear()` – modify the model year and price instance variables:
  - increase the model year by one; and
  - the price of cars with 2 doors go up 5% (these are probably sports cars); or
  - the price of all other cars and all motorcycles go up by 3%.

Write **constructors** for each class to initialize its instance variables. You do **not** need to write any toString() methods.

Note: it is not necessary to write a main program or any other classes/methods that create objects. **You must use proper object-oriented design techniques for your classes; use inheritance to avoid repeated or unnecessary code. Make all your instance variables private.**

Question 12, cont'd

Use this space to continue your answers to any earlier questions.
Make sure you clearly label all work that you want marked!