Introduction to Computer Science II (CSI 1101 A,B)
Midterm Examination

Instructor: Marcel Turcotte

February 2004, duration: 2 hours

Identification

Student name: _____

Student number: _____ Signature: _____

Instructions

1. This is a closed book examination;
2. No calculators or other aids are permitted;
3. Write comments and assumptions to get partial marks;
4. Beware, poor hand writing can affect grades;
5. Do not remove the staple holding the examination pages together;
6. Write your answers in the space provided. Use the backs of pages if necessary. You may **not** hand in additional pages;
7. Appendix A lists some of the methods of the classes `String` and `Character`.

Marking scheme

Question	Maximum	Result
1	18	
2	36	
3	12	
4	12	
5	16	
6	6	
Total	**100**	

Question 1 (18 marks)

This question is about a class to represent rational (fractional) numbers. For each of the following subquestions clearly label the necessary changes in the implementation that can be found on page 4.

1. (8 marks) Make all the necessary changes to the class `Rational` (found on page 4) so that it implements the interface `Comparable`.

```
public interface Comparable {

     // Compares this object with the specified object for order.
     // Returns a negative integer, zero, or a positive integer
     // as this object is less than, equal to, or greater than the
     // specified object.

    public int compareTo(Object o);
}
```

2. (6 marks) Implement the instance method `public boolean equals(Object o)` that returns `true` if this fraction and the one designated by `o` represent the same fraction (content equality).

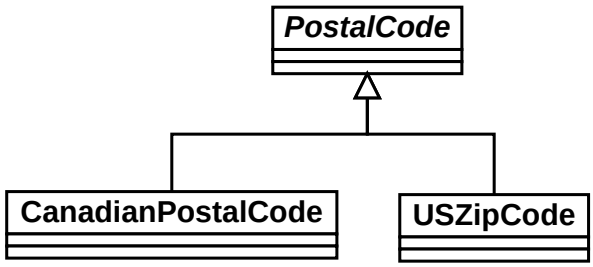
3. (4 marks) Add a method `public String toString()` that returns a `String` representation of this fraction with the numerator followed by the symbol "/", followed by the denominator. If the denominator is 1 then the method returns a `String` consisting of the numerator only.

```
public class Rational {
    // instance variables
    private int numerator;
    private int denominator;

    // constructor
    public Rational(int numerator) {
        this(numerator, 1);
    }
    public Rational(int numerator, int denominator) {
        if (denominator < 0) { // numerator stores the sign
            denominator = -1 * denominator;
            numerator = -1 * numerator;
        }
        this.numerator = numerator;
        this.denominator = denominator;
        reduce();
    }
    // instance method(s)
    public Rational plus(Rational other) {
        int newDenominator = denominator * other.denominator;
        int newNumerator = numerator * other.denominator;
        int newOtherNumerator = other.numerator * denominator;
        int sum = newNumerator + newOtherNumerator;
        return new Rational(sum, newDenominator);
    }
    // Transforms this number into its reduced form
    private void reduce() {
        if (numerator == 0) {
            denominator = 1;
        } else {
            int common = gcd(Math.abs(numerator), denominator);
            numerator = numerator/common;
            denominator = denominator/common;
        }
    }
    // Euclid's algorithm for calculating the greatest common divisor
    private int gcd(int a, int b) {
        while (a != b)
            if (a > b) {
                a = a - b;
            } else {
                b = b - a;
            }
        return a;
    }
}
```

Question 2 (36 marks)

The UML diagram below shows a hierarchy of classes to represent postal codes of different countries.



Knowing that:

- All postal codes have a method `getCode` that returns the code (of type `String`) represented by this instance;
- All postal codes have a method `isValid` that returns `true` if this code is valid and `false` otherwise;
- A Canadian postal code is valid if positions 0, 2 and 5 are letters, positions 1, 4 and 6 are digits, and the position 3 is a blank character;
- A valid US zip code consists of two letters, followed by a blank character, followed by 5 digits.

1. (30 marks) Write an implementation for the classes `PostalCode`, `CanadianPostalCode` and `USZipCode`. Make sure to include the instance variables and necessary constructors. Appendix A lists some of the methods of the classes `String` and `Character`.

Question 2 (continued)

Question 2 (continued)

2. Declare an array, called `codes`, to hold exactly 100 postal codes (3 marks).

3. Knowing that exactly `n` postal codes are currently stored in the left most cells of the array, write a `for` loop to count the number of valid codes (3 marks).

Question 3 (12 marks)

In the `ArrayStack` class below (similar to the one in assignment #4) write an instance method, `decreaseSize`, that will decrease the size of the array of `Objects`, `elems`, by `GROWTH_FACTOR` when appropriate. The size of the array must be decreased as soon as possible, without loosing an element. The minimum size of the array at any time is `DEFAULT_CAPACITY`.

```
public class ArrayStack {

    // constants
    public static final int DEFAULT_CAPACITY = 10;
    public static final int GROWTH_FACTOR = 2;

    // instance variables
    private Object[] elems; // contains the elements of this stack
    private int top = -1;   // designates the top element
                            // or -1 if this stack is empty
    // constructor
    public ArrayStack() {
        elems = new Object[DEFAULT_CAPACITY];
    }

    // returns true if this stack is empty;
    public boolean isEmpty() {
        return top == -1;
    }

    // returns and remove the top element of this stack.
    public Object pop() {
        // pre-conditions: ! isEmpty()

        // decrements top, then access the value
        Object saved = elems[top];
        // scrub the memory!
        elems[top--] = null;

        decreaseSize();

        return saved;
    }
```

Question 3 (continued)

```
    private void decreaseSize() {

    } // End of the method decreaseSize
} // End of the class ArrayStack
```

Question 4 (12 marks)

Complete the implementation of the instance methods `size()` and `swap()` within the class `LinkedStack` below. The method `size()` returns the number of elements that are currently stored into this stack. The method `swap` exchanges the first two elements (not the values); the first element becomes the second and the second element becomes the first. The method returns `false` if there are less than 2 elements in the list. You cannot use the methods `push` and `pop`, instead the links (references) must be manipulated.

```
public class LinkedStack {

    // Objects of the class Elem are used to store the elements of the stack

    private static class Elem {
        private Object value;
        private Elem next;
        Elem(Object value, Elem next) {
            this.value = value;
            this.next = next;
        }
    }

    // Instance variable

    private Elem topElem;

    public int size() {

        Elem _______ = _______;

        int count = 0;

        while (_________________) {

            count++;

            _________________;
        }

        return count;
    }
```

Question 4 (continued)

```
 public boolean swap() {

     // pre-condition(s)

     if (_______________________________________) {
         return false;
     }

     Elem first = _________________;

     Elem second = _________________;

     first.next = _________________;

     second.next = _________________;

     topElem = _________________;

     return true;
 }

 // The other instance methods, such as push and pop
 // would go here; but they cannot be used to answer
 // this question.

} // End of the class LinkedStack
```

Question 5 (16 marks)

Write a class (static) method that returns a copy of the stack given as an argument. For this question, there is an interface called `IntStack` and its implementation, a class called `MysteryIntStack`. The elements of the stack are of type `int`.

```
public interface IntStack {
    // returns true if this stack is empty
    public abstract boolean isEmpty();
    // pushes an element onto the top of this stack
    public abstract void push(int value);
    // removes and returns the top element
    public abstract int pop();
}
```

The class `MysteryIntStack` implements the interface IntStack. The constructor (`public MysteryIntStack()`) creates an empty stack. The implementation can hold an arbitrarily large number of elements. Make no other assumption about this implementation; in particular, you cannot assume that it uses an array or a linked-list.

For the class `Utils` below, write a class (static) method that returns a **new** stack that contains the same elements as the one designated by the parameter `in`, in the same order. Furthermore, the parameter `in` must not be changed (i.e. after the method call, `in` must contain the same elements, in the same order, as before the call).

```
public class Utils {
    public static IntStack copy(IntStack in) {
```

Question 5 (continued)

```
    } // End of the method copy
} // End of the class Utils
```

Question 6 (6 marks)

1. (2 marks) Give the postfix expression corresponding to the following infix expression.

$$(6 - 8 / 2) * (4 * 3 - (5 + 2))$$

2. (4 marks) Evaluate the following postfix expression using the algorithm seen in class, and in your lecture notes. Show the content of the stack **before** and **after** the evaluation of each subexpression.

$$12\ \ 6\ \ -\ \ 33\ \ 3\ \ /\ \ +$$

A Appendix

The class `String` contains the following methods.

char charAt(int pos): Returns the character at the specified index.

int length(): Returns the length of this string.

The class `Character` contains the following methods.

static boolean isDigit(char ch): Determines if the specified character is a digit.

static boolean isLetter(char ch): Determines if the specified character is a letter.

static boolean isWhitespace(char ch): Determines if the specified character is white space according to Java.

(blank space)