PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

MAY EXAMINATIONS 2002

CSC 108H1 S
St. George Campus

Duration — 3 hours

Aids allowed:
The Java API:
An Introduction for Students

PLEASE HAND IN

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

Last Name: ____________________

First Name: ____________________

Lecture Section: __________ **Instructor:** __________

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.)

This examination consists of 7 questions on 12 pages (including this one). *When you receive the signal to start, please make sure that your copy of the examination is complete.* 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.*

Comments are not required, although they may help us mark your answers. They may also get you part marks if you can't figure out how to write the code.

Write your student number at the bottom of pages 2-12 of this test.

1: ______/ 12

2: ______/ 14

3: ______/ 30

4: ______/ 6

5: ______/ 16

6: ______/ 14

7: ______/ 8

TOTAL: ______/100

Good Luck!

Question 1. [12 MARKS]

Part (a) [8 MARKS]

Write the methods less and more of class Twist (no constructor is needed).

```
/** Twist models a devious object
*/
public class Twist {
    // value is this Twist's favourite number.  Break tradition: make it public
    public int value= 0;

    // [4 marks]
    /** less: Takes a single parameter, int k.
     *  Decrease the value stored in k by 1, then return k.
     */









    // [4 marks]
    /** more: Takes a single parameter of type Twist, called curl.
     *  Increase the value stored in curl.value by 1, then return curl.
     */









    /** read through main, so you can predict the output
     */
    public static void main(String[] args) {
        Twist t1= new Twist(); Twist t2= new Twist();
        int k=5;
        t1.less(k); System.out.println(k);
        t1.more(t2); System.out.println(t1.value);
        k= t2.less(k);
        t1= t1.more(t2);
        System.out.println(k);
        System.out.println(t1.value);
    }
}
```

Part (b) [4 MARKS]

Write the output you expect from running Twist:

Question 2. [14 MARKS]

Recall assignment 4, where you wrote the Hangman program. In that assignment your program presented a String indication of how many characters were in a word, and the user guessed letters in the word. As the user guessed letters, your program placed them in the String showing what portion of the word had been guessed.

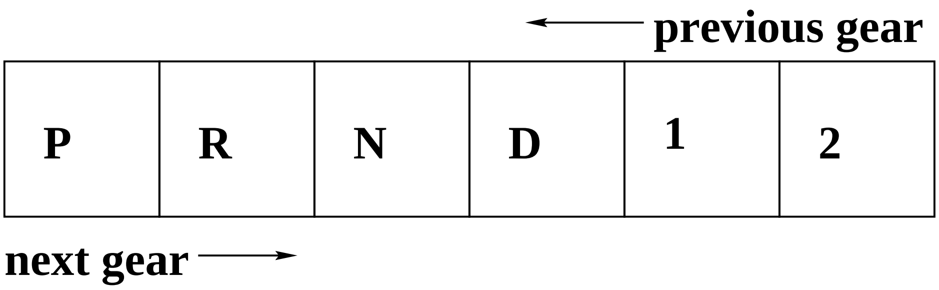
Complete the HangWord class below, to help the Hangman program. Use loops and conditionals where appropriate.

```
/** Game models a Hangman game.
*/
public class HangWord {
    // trueWord is the word to be guessed
    private String trueWord;
    // guessedWord contains * for unguessed characters, and already guessed characters
    private String guessedWord;

    // [6 marks]
    /** constructor: create a new HangWord with trueWord set to word.
     *  guessedWord should have the same length as trueWord, but contain only asterisks ("*" character).
     */
    public HangWord(String word) {









    }

    // [5 marks]
    /** buildGuess: Assume that s is a String of length 1.  For every position
     *  that s occurs in trueWord, place s in that position in guessedWord.
     */
    public void buildGuess(String s){









    }

    // [3 marks]
    /** isCorrect: return true if the user has guessed all characters in trueWord */
    public boolean isCorrect() {



    }
}
```

Question 3. [30 MARKS]

This problem is similar to the 3-way switch problem from tutorial preparation exercises.

You will complete the skeleton class `GearShift` (below) to simulate the gear shift in a car with an automatic transmission. A `GearShift`'s `currentGear` may be park, reverse, neutral, drive, first, or second. Choose a convenient way to store `currentGear`. A `GearShift` can shift from park to reverse, from reverse to either park or neutral, from neutral to either reverse or drive, from drive to either neutral or first, from first to either drive or second, and from second to first. We denote these shifts as being either to the next or previous gear in accordance with the following diagram:



Part (a) [5 MARKS]

Declare any instance variables you'll need for `Gearshift`, and write a `constructor` that creates a new `GearShift` with `currentGear` park (for safety reasons).

Part (b) [6 MARKS]

Write a method called `nextGear` that moves the gear shift from `currentGear` to the next gear (according to the diagram). For example, if the `currentGear` is park, `nextGear` will move it to reverse. If `currentGear` is second, this method should indicate to the user that this is impossible. If `currentGear` is drive, `nextGear` will move it to first.

Part (c) [6 MARKS]

Write a method called `previousGear` that moves the gear shift from `currentGear` to the previous gear (according to the diagram). For example, if `currentGear` is second, `previousGear` moves it to first. If `currentGear` is park, this method should indicate to the user that this is impossible. If `currentGear` iss neutral, `previousGear` will move it to reverse.

Part (d) [3 MARKS]

Write a `toString()` method that returns `currentGear` as a `String`.

```
public class GearShift { // continued on the next page...
```

```
// GearShift continued from the previous page...
```

```
// end of GearShift
}
```

Part (e) [10 MARKS]

Describe five appropriate test cases to run on your `GearShift` class. Give specific examples, stating the current gear, the method you request, and the expected result. State why each is an appropriate test.

Question 4. [6 MARKS]

Recall the `StudentRecord` part of assignment 5, where you were asked to write a method to merge two sets of marks. Each mark was either a decimal number represented as a `String`, for example "65.7", or the empty `String` "" (to represent a missing mark). The `mergeRecords` method in this exam question only merges sets of marks, unlike the assignment where you also had to deal with names and student numbers.

Complete the `mergeRecords` method of the class `StudentRecord`, below (the other methods are omitted for brevity). The method has a single parameter of type `StudentRecord` called `other`, and merges the `String` array `marks` with `other.marks`, according to the following rules: If a mark is missing in one array, the corresponding mark from the other array is used (you may assume that they are not both missing). If both marks exist, take the higher one. The method `mergeRecords` returns an array of `String`.

```
/** An abbreviated StudentRecord class
 */
public class StudentRecord {
    // 15 marks are stored as Strings representing decimal numbers
    private String[] marks= new String[15];

    // constructor and lots of methods omitted...

    /** mergeRecords: return a String array containing the
     *  merged marks from this StudentRecord and other, according
     *  to the rules stated above.  You may assume that there is
     *  at least one valid mark in the pair marks[i], other.marks[i],
     *  for each 0 <= i < 15.
     */
    public String[] mergeRecords(StudentRecord other) {







    } // end of method mergeRecords
}
```

Question 5. [16 MARKS]

Read the skeleton of class `Paragraph`, complete the `constructor`, plus the methods `loadParagraph`, `countVowels`, and `countVowelsInParagraph`.

```
/** Paragraph reports statistics on some lines of text.
 */
import java.io.*;
public class Paragraph {
  // myParagraph holds the lines of text.
  private static Vector myParagraph= new Vector();
  // br is attached to the keyboard when the class is loaded
  private BufferedReader br= new BufferedReader(new InputStreamReader(System.in));

  // [4 marks]
  /** loadParagraph: use br to read input from user until they type QUIT
   *  on a line by itself.  Store each line (except QUIT) in myParagraph.
   */
  public void loadParagraph() {







  }

  // [1 marks]
  /** constructor: use loadParagraph() to create a Paragraph
   *  with myParagraph containing lines typed by user
   */
  public Paragraph() {


  }

  // [5 marks]
  /** countVowels: return the number of vowels in line.
   */
  public int countVowels(String line){















  }

  // class Paragraph continued on next page
```

```
  // [6 marks]
  /** countVowelsInParagraph: return number of vowels in myParagraph.
   */
  public int countVowelsInParagraph() {











  }
} // end of class Paragraph
```

Question 6. [14 MARKS]

We will say positive integers `n1` and `n2` are **coprime** if `gcd(n1, n2)` (defined below) returns 1.

```
/** gcd: return the greatest common divisor of num1 and num2.
 *  require: num1 and num2 are positive integers.
 */
public static int gcd(int num1, int num2) {
  while ((num1 % num2) != 0) {
    int tmp= num2;
    num2= num1 % num2;
    num1= tmp;
  }
  return num2;
}
```

Part (a) [2 MARKS]

Use `gcd(int n1, int n2)` to write a boolean method `coprime` that returns `true` if `n1` and `n2` are coprime, and `false` otherwise.

```
public boolean coprime(int n1, int n2) {

}
```

Part (b) [7 MARKS]

Assume that `numbers` is an array of positive integers. Write a boolean method `pairwiseCoprime` that returns `true` if `numbers[i]` and `numbers[j]` are coprime for **every** pair of indices `i, j` between 0 and `numbers.length - 1`, provided `i` is different from `j`. Return `false` otherwise.

```
public boolean pairwiseCoprime(int[] numbers) {

}
```

Part (c) [5 MARKS]

Assume that `numbers` is an array of positive integers. Write a boolean method `neighbourCommon` that returns `true` if `gcd(numbers[i], numbers[i+1])` is greater than 1 whenever `0 <= i < numbers.length-1`. Return `false` otherwise.

```
public boolean neighbourCommon(int[] numbers) {



}
```

Question 7. [8 MARKS]

Please read through the classes Animal, Mammal, Primate, and TestsBeasts. When you are finished, write down the output you would expect to see displayed after running TestBeasts. Note that all four classes compile, and TestBeasts runs without errors.

```java
/** Animal model an animal */
public class Animal {
  // this animal's name
  private String name;

  /** constructor: Create an Animal with name n
   */
  public Animal(String n) {
    name= n;
  }

  /** toString: represent this animal
   *  as a String.
   */
  public String toString() {
    return name;
  }

}

/** Mammal models a mammal */
public class Mammal extends Animal {
  // body temperature, in degrees celsius
  private double bodyTemperature;

  /** constructor: Create a mammal with name n,
   *  and body temperature t.
   */
  public Mammal(String n, double t) {
    super(n);
    bodyTemperature= t;
  }

  /** toString: represent this mammal
   *  as a String.
   */
  public String toString() {
    String result= super.toString();
    result = result + "\nBody temperature: " +
        bodyTemperature;
    return result;
  }

}
```

```java
/** Primate models one of our closer relatives */
public class Primate extends Mammal {
  // the highest number this primate can count to...
  private int upperBound;

  /** constructor: Create a primate with name n,
   *  body temperature t, and upperBound u.
   */
  public Primate(String n, double t, int u) {
    super(n, t);
    upperBound= u;
  }

  /** toString: represent this primate
   *  with a String.
   */
  public String toString() {
    String result= super.toString();
    result = result + "\nCounts to: " + upperBound;
    return result;
  }

}

/** TestBeasts test drives some beasts */
public class TestBeasts {
  public static void main(String[] args) {
    Animal[] bestiary= new Animal[4];
    Animal tmpAnimal;
    bestiary[0]= new Animal("gecko");
    bestiary[1]= new Mammal("platypus", 36.3);
    bestiary[2]= ((Animal) new Mammal(
        "gerbil", 38.5));
    bestiary[3]= ((Mammal) new Primate(
        "lemur", 38.4, 3));
    tmpAnimal= bestiary[0];
    bestiary[0]= (Animal) bestiary[3];
    bestiary[3]= tmpAnimal;

    for (int i= 0; i != bestiary.length; i++) {
      System.out.println(bestiary[i]);
      System.out.println();
    }
  }
}
```

Write expected output below here: