THE UNIVERSITY OF WESTERN AUSTRALIA

Computer Science and Software Engineering

SEMESTER 1, 2016 EXAMINATIONS

CITS1001
Object-oriented Programming and Software Engineering

FAMILY NAME:______________________ GIVEN NAMES: ______________________

STUDENT ID: | | | | | | | | | SIGNATURE:______________________

This Paper Contains: **20** pages **(including title page)**
Time allowed: **2:00 hours (including reading time)**

INSTRUCTIONS:
Answer all questions. The paper contains eight questions, each worth ten marks.
Write your answers in the spaces provided on this question paper.
No other paper will be accepted for the submission of answers.
Do not write in this space.

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Supervisors Only - Student left at:

This page has been left intentionally blank

This page has been left intentionally blank

Class structure

1a) Write a Java class `Country` to represent a country that is a member of the European Union. The class should have

- three instance variables that capture a country's name, the year it joined the EU, and whether it uses the Euro as its currency;
- a constructor that sets up each of these variables (no error-checking is required);
- accessor methods for each of these variables; and
- a mutator method for the Euro variable. **(4 marks)**

1b) Write a Java class `EU` to represent the member countries of the European Union. The class should have

- one instance variable that contains information about the member countries;
- an accessor method that takes an integer and that returns the corresponding `Country` object;
- a method `newMember` that takes a `Country` object and updates the instance variable appropriately; and
- a method `needFX` that takes two `Country` objects and returns true iff you need to change currency when moving from one to the other. **(6 marks)**

Use this space and the following page for your answer to this question

Numbers

2) Write the three methods marked `TODO` below.

```
public class Rectangle
{
    // bottom-left and top-right corners of a rectangle
    // with sides are parallel to the axes
    private double blX, blY, trX, trY;

    public Rectangle(double x1, double y1, double x2, double y2)
                throws Exception
    {
        if (trX <= blX || trY <= blY)
           throw new Exception("invalid coordinates");
        blX = x1;
        blY = y1;
        trX = x2;
        trY = y2;
    }

    // returns the area of the rectangle
    public double area()
    {
        // TODO                                          (2 marks)
    }

    // returns true iff the point x,y is inside the rectangle
    public boolean inside(double x, double y)
    {
        // TODO                                          (3 marks)
    }

    // moves the top-right corner to form the largest square
    // that fits inside the original rectangle
    public void largestSquare()
    {
        // TODO                                          (5 marks)
    }
}
```

Use this space and the following page for your answer to this question

Booleans

3a) Using only conditionals and relational operators, write the method `smallestPositive` that takes two numbers and returns the smallest positive one. If neither argument is positive, it returns `0`.
For example, `smallestPositive(7,4)` and `smallestPositive(4,7)` both return `4`, `smallestPositive(7,-4)` and `smallestPositive(-4,7)` both return `7`, and `smallestPositive(-7,-4)` returns `0`. **(6 marks)**

```
// returns the smallest positive of x and y;
// or 0 for two non-positive arguments
public int smallestPositive (int x, int y)
```

3b) Using only logical (Boolean) operators, write the method `mixture` that takes three Booleans and returns `true` if and only if they contain at least one `true`, and also at least one `false`.
For example, `mixture(true,false,false)` and `mixture(false,true,true)` both return `true`, but `mixture(true,true,true)` returns `false`. **(4 marks)**

```
// returns true iff x,y,z are a mixture of true and false
public boolean mixture(boolean x, boolean y, boolean z)
```

Use this space and the following page for your answer to this question

Arrays

4a) Write the method `middle` that takes an array `a` and returns a new array holding the middle third of `a`. You may assume that the length of `a` is a multiple of 3.
For example, `middle({5,4,1,6,3,2})` returns `{1,6}`. **(4 marks)**

```
// returns the middle third of a
public static int[] middle(int[] a)
```

4b) Write the method `reversed` that returns `true` if and only if the arrays `a` and `b` contain exactly the same elements, but in reversed order.
For example, `reversed({3,1}, {1,3})` returns `true`, but
`reversed({3,1}, {2,3})` and `reversed({3,1}, {1,1,3}`
both return `false`. **(6 marks)**

```
// returns true iff a and b contain the same elements, reversed
public static boolean reversed(int[] a, int[] b)
```

Use this space and the following page for your answer to this question

Display

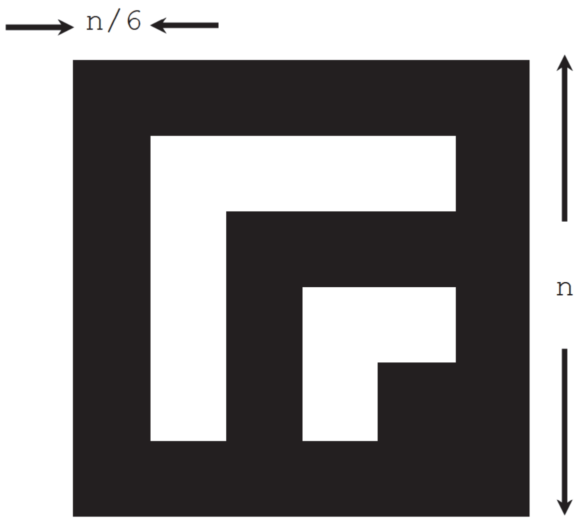
5) For the following questions, you may assume that both `java.awt.Color` and `SimpleCanvas` have been imported.

a) Write the method `flip` that takes a colour as its argument. If the argument is white, it returns black; if the argument is black, it returns white; if the argument is anything else, it throws an `IllegalArgumentException`. **(3 marks)**

```
// returns the opposite colour to col
public Color flip(Color col)
```

b) Use `flip` and the `SimpleCanvas` method `drawSq` below to write the method `drawNoland` that draws the flag of Noland, as illustrated below. The flag must be square with side `n`, and the width of each stripe must be one-sixth of `n`. **(7 marks)**

```
// draws a square of side s and colour c on its canvas,
// with its top-left corner at x,y
public void drawSq(int x, int y, int s, Color c)
```



```
// draws the flag of Noland with side n
public void drawNoland(int n)
```

Use this space and the following page for your answer to this question

Data compression

6) One obvious way to compress binary data is to count the 0s and 1s, and to represent sequences of one digit as a single number. Thus a String sequence like "00100111" would be compressed to "2,1,2,3". The first number in the compressed sequence is always the number of initial 0s, so a sequence like "1101111" would be compressed to "0,2,1,4".

a) Write the method `compress` that takes a String `s` containing only 0s and 1s, and returns a compressed version of `s`. **(5 marks)**

```
// returns a compressed version of s
public static String compress(String s)
```

b) Write the method `uncompress` that takes a String `s` containing only digits, and returns an uncompressed version of `s`. **(5 marks)**

```
// returns an uncompressed version of s
public static String uncompress(String s)
```

Use this space and the following page for your answer to this question

Sorting

7) One of the fastest known sorting algorithms is *Heapsort*. The key phase of this algorithm reorders an array `a` of integers so that two properties hold for all indices `k`:

`a[k] >= a[2k+1]` and `a[k] >= a[2k+2]`

This phase of Heapsort is implemented by the method `heapify`.

```
// heapify(a) turns a into a heap in-place
public void heapify(int[] a)
{
  for (int root = a.length / 2 - 1; root >= 0; root--)
  {
    boolean done = false;
    while (2 * root + 1 < a.length && !done)
    {
      int swap = root;
      if (a[swap] < a[2 * root + 1])
         swap = 2 * root + 1;
      if (2 * root + 2 < a.length && a[swap] < a[2 * root + 2])
         swap = 2 * root + 2;
      if (swap == root)
         done = true;
      else
      {
         int temp = a[root];
         a[root] = a[swap];
         a[swap] = temp;
         root = swap;
      }
    }
  }
}
```

a) Show how the application `heapify({1,2,3,4,5,6,7,8,9})` is processed.
In particular, show the state of `a` every time it is changed by `heapify`. **(7 marks)**

b) For what data will `heapify` be especially fast? **(3 marks)**

Use this space and the following page for your answer to this question

Recursion

8a) What is the role of the *base cases* in a recursive Java method? **(2 marks)**

8b) Exponentiation can be implemented efficiently using the following equations.

$$
\begin{aligned}
x^0 &= 1 \\
x^{2n} &= (x^2)^n \\
x^{2n+1} &= x^{2n} * x
\end{aligned}
$$

Write the recursive method `exp` that calculates x^n using this algorithm. **(5 marks)**

```
// returns x raised to the power n
public static double exp(double x, int n)
```

8c) Show the sequence of method calls that your definition generates for the invocation `exp(3,6)`. **(3 marks)**

Use this space and the following page for your answer to this question

END OF PAPER