ASSIGNMENT 1
Binary and Math

COMP-202, Summer 2011

Due: Wednesday, May 18, 2011 (23:30)

You must do this assignment individually and, unless otherwise specified, you should follow all the general instructions and regulations for assignments. Graders have the discretion to deduct up to 10% of the value of this assignment for deviations from the general instructions and regulations.

Part 1:	0 points
Part 2, Question 1:	30 points
Part 2, Question 2:	20 points
Part 2, Question 3:	20 points
Part 2, Question 4:	15 points
Part 2, Question 5:	15 points
Part 2, Question 6:	5 points
	100 points total

Part 1 (0 points): Warm-up

Do **NOT** *submit this part, as it will not be graded. However, doing these exercises might help you to do the second part of the assignment, which will be graded. If you have difficulties with the questions of Part 1, then you can consult Dan or the TAs during their office hours; they can help you and work with you through the warm-up questions.*

Look at the old course website: `http://www.cs.mcgill.ca/~cs202/2011-01/assignments.html` under Assignment 0 and Assignment 1. You should try to do some of the questions on these. Of particular significance are:

- Assignment 0 : Warm up question number 1
- Assignment 0: Warm up question 2
- Assignment 0: Graded question 3
- Assignment 0: Graded question 4
- Assignment 0: Graded question 5
- Assignment 1: Warm up question 3
- Assignment 1: Warm up question 4
- Assignment 1: Graded Question 1
- Assignment 1: Graded Question 2
- Assignment 1: Graded Question 3

In addition, you should read over the general assignment guidelines posted on the course website at `http://www.cs.mcgill.ca/~dpomer/comp202/summer2011/assignments/instructions.pdf`. Some of these will not be enforced immediately, particularly the style-related ones, but you should always keep them in mind from the beginning. (Bad habits are hard to break!)

Part 2

The questions in this part of the assignment will be graded.

Question 1: Binary and other counting systems (30 points) 1.

Convert the following numbers from base 10 to a)binary, b)base-3 and c)hexadecimal (base 16)

(a) 46

(b) 642

(c) 9011

(d) 1651

2. Convert the following numbers from their base (specified by the subscript) to base-10. For example 524_7 means the number 524 in base-7.

    (a) 516_9

    (b) 1110011_2

    (c) $a5b_{16}$

    (d) 1110000111_2

3. Perform the following arithmetic in other bases. You should do this first by performing the operation in the given base. Then, to check your answer, try converting the numbers to base-10, adding them "normally" and converting the number back to binary.

    (a) $1100001011_2 + 1001111101111_2 =?$

    (b) $221210012_3 - 112111002_3$

    (c) $1011_2 * 1111_2$

    For this question, you should put your answers in a nicely formatted, easy to read txt file called CountingSystems.txt However, if you wish to show your work, you may also hand in a pdf file (in addition to the txt)

Question 2: Using Variables and Expressions to Approximate sin(x) (20 points)

A Taylor series is a way to approximate a complicated mathematical function using a polynomial. Polynomials are easy for a computer to evaluate because they only involve addition/subtraction and multiplication. Compared to functions such as $log(x)$ or $sin(x)$, this is much simpler to put into a computer.

In this question, you will write a program that evaluates $sin(x)$ for a few different values of x using the Taylor series approximation for $sin(x)$. In the later questions, you will expand upon this program to make it more general and re-usable. You should hand in all questions as this will be necessary in the case that your later questions don't work correctly.

$$sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \approx x - \frac{x^3}{3!} + \frac{x^5}{5!}$$

Write a Java program to evaluate $sin(x)$ using the above formula (the abbreviated part) for values x = 0,.5,1,1.5,2.

The beginning of your output should look roughly as follows (don't worry about exact spacing but the line numbering should be the same)

```
The approximation for sin(0) is 0.0.
The approximation for sin(.5) is .4794271.
```

Put your program into a class (and file) called SinApproximation (and SinApproximation.java).

For obvious reasons, you may not use any methods in the Math library. Also you will probably find it very helpful to copy/paste here.

Question 3: Using Methods to Reduce Code (20 points)

In this question, you will "refactor" the code written in Question2 in order to make it more readable and re-usable.

First, write a method `public static double mySin(double x)` which takes as input a double and returns a double, calculated by the $sin(x)$ approximation as in the previous question.

One way to do this is to take the code written in the previous question for a specific value (for example .5) and replace all occurrences of the specific value with the variable `x`. You can then leave your answer in terms of `x` and return that value.

Now, change your main method from Question2 so that it just calls the mySin method that you have just written several times. It should then print the result for the same 5 values of x as in the previous part. You can either print the result of $mySin(x)$ directly or store the result in a variable, but there should be no math taking place in the main method other than calling the `mySin` method.

Put your program into a class (and file) called WithMethods (and WithMethods.java)

Question 4: Approximating $cos(x)$ (15 points)

One can approximate $cos(x)$ in a similar way. The Taylor formula to do this is:

$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$$

Add to your program in the previous part a method called `myCos(x)` . The method should (like `mySin(x)` method) take as input a double and return a double. It should compute the double based on the Taylor formula above.

Add to your main method several calls to `myCos(x)` . One good way to check that you have coded correctly is that `myCos(x) * myCos(x) + mySin(x) * mySin(x)` should be close to 1. It will not be exactly one since it is an approximation. The approximation works better with values of x close to 0.

Add your method to the class `WithMethods` in the file WithMethods.java

This method will be very similar to the `mySin` method, except with a different formula applied.

Question 5: Using library methods (15 points)

There are many methods that come with your installation of Java. Some of these are in the `Math` library. You can find out information about these at: `http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Math.html` . Because a) these methods are part of the `Math` library, and defined in a class called `Math` and b)you are writing your methods in a class other than the `Math` class, you must call them by prefixing the name of the method with `Math.` (including the period).

For example, to call the first method defined at the link, `abs(double a)` , you could write:

```
double d = Math.abs(-5); //d now equals 5.0
```

In this question, you will compare the methods `mySin(x)` to the sin method definted in the `Math` library. For every value of x from 0 to 1, increasing by .1, you should output the following in this order. It is important to stick to the formatting as carefully as possible to ensure your assignment is marked correctly.

1. The value of x
2. The result of calling mySin(x)
3. The result of calling Math.sin(x)
4. The absolute value of the difference between the above two results. (You can use the abs library method here if you like, or you can write your own)

Each value should have text before indicating what it is.

For example, one line could look like:

```
x=0 mySin(x)=0.0 Math.sin(x)=0.0 Difference=0.0
```

You may use a loop or copy/paste for this question. (If you use a loop, you will receive bonus points as below but make sure that you have all the right ranges.)

Put your program into a class called `MySinComparison`

Question 6: Bonus question (5 points)
Write question 5 above using a loop.

What To Submit

`confession.txt` - You should write in this file any information that you think is useful for the TA to mark the assignment. This should include things you were not sure of as well as parts of your code that you don't think it will work. Of course, like a confession, this will draw the TA's attention to the part of your code that doesn't work, but he/she will probably be more lenient than if he/she has to spend a lot of time looking for your error. It demonstrates that even though you couldn't solve the problem, you understand roughly what is going on.

```
CountingSystems.txt
CountingSystems.pdf (optional)
SinApproximation.java
WithMethods.java
MySinComparison.java
```