Assignment 2

Due: Feb 13, 2014 at 11:55PM.

You may discuss any of the assignments with your classmates (or anyone else) but all work for all assignments must be entirely your own. Any sharing or copying of assignments will be considered cheating.

Problem 1 (30 points)

Write a program that reads from the user an integer and prints out the name of the month if the number is between 1 and 12, or prints out an error message if the number is outside of the 1-12 range. The names of months printed should be "January" for 1, "February" for 2, and so on. The error message should be "This is not one of the months in our calendar."

You have to use the switch statement in your implementation.

Problem 2 (35 points)

Write a program that approximates the value of $\pi$ using the following formula:

$$\pi \approx 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots + \frac{(-1)^{i+1}}{2i-1}\right).$$

Your program should compute three different approximations for $i = 1000$, $i = 10000$, and $i = 100000$.

This program does not use any input from the user. It outputs to the standard output three statements:

The first approximation to PI is: ...
The second approximation to PI is: ...
The third approximation to PI is: ...
The value of constant Math.PI is: ...

The ... in each of the above statements should be replaced by one of the three values computed by your program and the value provided by Math.PI constant. You should use formatted output (System.out.printf statements) to align all the colons (:) in the printed lines and to display the numbers in the fields of width 30 with 28 digits after the decimal point.

HINT 1: the three approximations that you compute should be getting closer and closer to the value stored in Math.PI constant.
HINT 2: You need to use a loop to compute each approximation.

Problem 3 (35 points)

Write a program that reads as input from the user a single positive integer, call it $n$, and then uses $n$ to compute the next number in the sequence as follows:

- if $n$ is even, the next number is $n$ divided by 2 (integer division),
- if $n$ is odd, the next number is $n$ multiplied by 3 plus 1.

The first approximation to PI is: ...
The second approximation to PI is: ...
The third approximation to PI is: ...
The value of constant Math.PI is: ...

The ... in each of the above statements should be replaced by one of the three values computed by your program and the value provided by Math.PI constant. You should use formatted output (System.out.printf statements) to align all the colons (:) in the printed lines and to display the numbers in the fields of width 30 with 28 digits after the decimal point.

HINT 1: the three approximations that you compute should be getting closer and closer to the value stored in Math.PI constant.
HINT 2: You need to use a loop to compute each approximation.
The next number is computed based on the previous value and so on. For example, if \( n \) starts at 11, the sequence is 11, 34, 17, 52, 26, 13, 40, 20, 10, ... .

This algorithm produces an infinite sequence of numbers. Compute the first hundred terms of the sequence and print the number to standard output. The numbers should be separated by a comma and a single space. Run your program several times using different input numbers. What do you notice about the sequences? (Answer this question in comments below the code for your program.)

**Grading**

**Does the program compile?** If not, you will lose all the points for that problem.

**Is the program properly documented?** (worth 30% of each problem)

Proper documentation includes:

- preamble with the name of the author, date of creation and brief description of the program;
- appropriately chosen variable names, i.e., descriptive names;
- comments inside the code describing steps need to be taken to accomplish the goal of the program;
- appropriate formatting, indentation and use of white space to make the code readable.

Remember that the code is read by humans and it should be easy to read for people who were not involved in its development.

**Is the program well developed?** (worth 40% of each problem) Make sure you create variables of appropriate types, use control statements (conditionals and loops) that are appropriate for the task, accomplish your task in a well designed and simple way (not a convoluted algorithm that happens to produce the correct output for some unknown reason). You should also design a friendly and informative user interface.

**Is the program correct?** (worth 30% of each problem), Make sure that your program produces valid results that follow the specification of the problem every time it is run. At this point you can assume a "well behaved user" who enters the type of data that you request. If the program is not completely correct, you get credit proportional to how well it is developed and how close you got it to the completely correct code.

**What and how to submit?**

You should submit three source code files combined into a single zip file to NYU Classes. Do not submit all the files that Eclipse creates, just the source code files that have .java extensions. Name your classes Problem1, Problem2 and Problem3 (this means that your files are going to be Problem1.java, Problem2.java and Problem3.java).

**Questions**

Post any questions you have regarding this assignment to NYU Classes Forums under the Assignment 2 topic. Do not post code to the forums.