**Reviewing Assignment**

Lab Assignment 2

|  |  |
| --- | --- |
| Started: | Sep 29, 2014 9:12 AM |
| Finished: | Oct 8, 2014 1:42 PM |

 1 of 1

**Lab Assignment 2     Total Grade: 20   (of possible 20 points)**

**Score: 20   (of possible 20 points)**

**Assignment 2 - Arithmetic**

Select one option from below.  All (both) options are worth the same number of points.  The more advanced option(s) are provided for students who find the basic one too easy and want more of a challenge.   Make sure you have read and understood

* both ***modules A*** and ***B*** this week, and
* ***module 2R - Lab Homework Requirements***

before submitting this assignment. Hand in only one program, please.

OPTION A (Basic):  Some Easy Arithmetic

*Every programming assignment is as much a test of****English language comprehension****as it is a test of programming or mathematical skills.  This week, the  explanations of the formulae below are given clearly in plain English if you read carefully.  However, if there is any question about what is being asked, you are urged to ask for clarification in the public forums.*

Understand the Application

You will create two **int** variables into which you will store

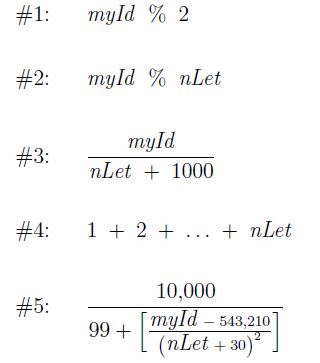
* Your Foothill College ***student ID.*** (This is normally an 8-digit number;  don't confuse it with your social security # or a password, which it is not.)
* The number of letters in your ***family (last) name***.

In order to receive any credit for this assignment, these two values must match what I have for you on my class roster.

Your program will compute some values based on these two numbers, so each student will have a unique output.

The Program Spec

The following five expressions assume that you have stored your student ID into the variable ***myId***, and the number of letters in your last (family) name into the variable***nLet***.  You can manually enter both of these using ***assignment statements*** in your program source.  No user input allowed.  Compute the following five values:



Write a program that computes and displays these five values.   Your run should look something like this (although the values will differ for each student):

/\* -------------------------- Run ----------------------

My last (family) name is Ying-Kam

My Student ID is 99999998

The number of characters in my last name is 8

Expression #1 ------------ : 0

Expression #2 ------------ : 6

Expression #3 ------------ : ...

...

------------------------------------------------------- \*/

Note that the first three lines are output from your program, not added after the run.  This should all be done in one run of a single program, not several runs.

Here are some tips and ***REQUIREMENTS***:

1. The "..." is called an "ellipsis" and means "and so forth."  So, in expression **#4** if your ***nLet*** is 8, the expression **1 + 2 + ... + *nLet***  really means 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8.  Since we have not covered loops, just use this latter expression to compute result in expression **#4**.
2. Do not try to take short cuts. For expression **#4**, I want to see every number in the computation.  As an example, in hint #1 directly above, you see that I listed the entire addition when describing the expression **1 + 2 + ... + *nLet***, so your program should also use the full expression. Remember that long source code lines (> 80 chars) are illegal according to my style rules.
3. Your personal info is not supposed to be entered by the user at run-time.  You should make assignments to the variables in your program.
4. Only one run, please, which means you must produce all **five** answers in your program in a single source file.
5. Expressions **#1**, **#2** and **#4** should use ordinary***int arithmetic***, which means using the **int** variables without any special tools.  However, expressions **#3** and **#5**are meant to display the***full double accuracy*** that the divisions will produce,  You should take appropriate action to do that.
6. Use as few variables as possible. You can, for example, use one **intResult** variable for the three int expressions and a second **doubleResult** variable for the two double expressions.  However, don't use five separate result variables.
7. As you see in the sample run, the first thing your program needs to do is print out your ***student ID***, your ***last (family) name***, and the number of characters, including special symbols, if any which you are using as a basis for these computations.
8. Do not use any power or exponential methods to compute #5.  First, we haven't had that yet, and second, it is inefficient to use a **pow()** method to compute small integral powers likes squares or cubes. Instead, use multiplication to compute the power(s) you need.

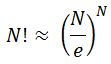
OPTION B-1 (Intermediate):  Stirling's Formula

This is for those of you who are already programmers and want something a little more challenging than the above option.  If you do it, please make sure you know my style rule for indenting*for loops*, *if statements* and *continuation lines*.

***Stirling's Formula*** is an approximation for ***N-factorial***, i.e., ***N!***,  that is increasingly accurate as the number ***N*** gets large.  ***Stirling's Approximation*** is:

https://myetudes.org/access/content/user/milo17780/Pics%20for%20All%20Classes/stirling_1.jpg

This is very important in physics and chemistry because, for very large ***N***, one cannot use computers (or even pen and paper) to compute ***N!*** directly.   When ***N***is on the order of ***Avogadro's Number*** (about **1023**) it is called a ***large number***.  In statistical mechanics, we often have even *larger* numbers, called **v*ery large numbers****,* the result of raising a number like **10** to a power which is a***large number***. (I'd use an exclamation point here for emphasis, but that would be abusive, in this context).  This approximation is excellent for ***large numbers*** and fantastically spectacular for ***very large numbers***.  In fact, for very large numbers we can actually ignore the radical because as physicists know, if you multiply a ***very large number*** by just a ***large number***, the ***very large number*** is unchanged!.  Thus we get a simpler expression that is sometimes used in place of Stirling's formula:



In this form, we can see how the approximation works: we are replacing each integer in the factorial by ***N/*e**, which acts as the representative value.  It is interesting that, by shrinking ***N*** by a factor of **e**, we get just the right number to represent all the factors.  When using ***large*** and ***very large numbers***, we are justified in simplifying our lives by taking the *natural log* of both sides, resulting in the following famous formula.  I call it ***Stirling's Log Approximation***:

https://myetudes.org/access/content/user/milo17780/Pics%20for%20All%20Classes/stirling_3.jpg

If you have had some calculus, you can basically give a back-of-the-envelope "proof" of this approximation in about 30 seconds by looking at the ***Riemann sum*** at intervals of size 1 of the function**log(*x)*** from ***1*** to ***N***.  These results are ever so helpful when we are doing simulations of heat exchange between solids or dust clouds forming stars in space.

Enough discussion.  Let's get to the assignment.  Write two programs (**main()**s):

1. **Program 1:**In a table, display ***N!*** for ***N*** = **1** to **30**, then display and compare each result with ***Stirling's Approximation***.  Compute the error in % (absolute value or actual)  for each entry in this table.
2. **Program 2:** In a separate table display **log *N!*** for ***N*** = **100** to **130** and compare with ***Stirling's Log Approximation.***  Again, compute the error.  Can you see the advantage of  using the **log**?

* *Have your program print out your****family name****and****student ID****at the start of the program, using simple output statements.  You will only receive credit if these are included in the run.*

There is no input for this program.  Also, submit all your output and both **main()**s in a single attachment, as usual.

/\* ------------- RUN 1 (OUTPUT) ----------------------

n n! Stirling's Approx Error

-------------------------------------------

1 1.0000E0 9.2214E-1 7.79%.

2 2.0000E0 1.9190E0 4.05%.

3 6.0000E0 5.8362E0 2.73%.

4 2.4000E1 2.3506E1 2.06%.

5 1.2000E2 1.1802E2 1.65%.

6 7.2000E2 7.1008E2 1.38%.

7 5.0400E3 4.9804E3 1.18%.

8 4.0320E4 3.9902E4 1.04%.

9 3.6288E5 3.5954E5 0.92%.

10 3.6288E6 3.5987E6 0.83%.

11 3.9917E7 3.9616E7 0.75%.

12 4.7900E8 4.7569E8 0.69%.

13 6.2270E9 6.1872E9 0.64%.

14 8.7178E10 8.6661E10 0.59%.

15 1.3077E12 1.3004E12 0.55%.

16 2.0923E13 2.0814E13 0.52%.

...

--------------------- Run #2 Stirling's Log Approximation -----

n log n! nLog(n) - n Error

-------------------------------------------

100 3.6374E2 3.6052E2 0.89%.

101 3.6835E2 3.6513E2 0.88%.

102 3.7298E2 3.6975E2 0.87%.

103 3.7761E2 3.7438E2 0.86%.

104 3.8226E2 3.7902E2 0.85%.

105 3.8691E2 3.8367E2 0.84%.

106 3.9158E2 3.8832E2 0.83%.

107 3.9625E2 3.9299E2 0.82%.

108 4.0093E2 3.9767E2 0.81%.

109 4.0562E2 4.0236E2 0.81%.

110 4.1032E2 4.0705E2 0.8%.

111 4.1503E2 4.1176E2 0.79%.

112 4.1975E2 4.1647E2 0.78%.

113 4.2448E2 4.2119E2 0.77%.

114 4.2921E2 4.2593E2 0.77%.

...

--------------------------------------------------- \*/

**Answer**

* text/plain[foothillAssignment2.txt](https://myetudes.org/access/mneme/content/private/mneme/cff3240c-b51c-41f6-80dc-4db4530bdd05/submissions/14605268/12ba0c82-5ce3-45c3-00c5-350a86a9d6b9/foothillAssignment2.txt)

[[https://myetudes.org/ambrosia_library/icons/collapse.gif](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/14605268/list) Model Answer](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/14605268/list)

/\* CS 1A Lab 2

 \* Instructor Solution

 \*/

public class Foothill

{

   public static void main(String[] args)

   {

      int resultInt;

      double resultDub;

      int studID, numLet;

      studID = 11111111;

      numLet = 3;

      // print header

      System.out.println("My last (family) name is Kam");

      System.out.println("My Student ID is " + studID);

      System.out.println("The number of characters in my last name is "

         + numLet);

      resultInt = studID % 2;

      System.out.println("\nExpression #1 ------------ : "

         + resultInt);

      resultInt = studID % numLet;

      System.out.println("\nExpression #2 ------------ : "

         + resultInt);

      resultDub = (double)studID /  (numLet + 1000.);

      System.out.println("\nExpression #3 ------------ : "

         + resultDub);

      resultInt = 1 + 2 + 3;

      System.out.println("\nExpression #4 ------------ : "

         + resultInt);

      resultDub = 10000. /

         (

            99 + (studID - 543210.) / ( (numLet + 30.) \* (numLet + 30.) )

         ) ;

      System.out.println("\nExpression #5 ------------ : "

         + resultDub);

   }

}

/\* ------------------- run -------------------------------

 \*

My Student ID is 11111111

The number of characters in my last name is 3

Expression #1 ------------ : 1

Expression #2 ------------ : 2

Expression #3 ------------ : 11077.877367896312

Expression #4 ------------ : 6

Expression #5 ------------ : 1.0200724785381996

------------------------------------------------------- \*/

**Comments**

Great work, Dmitri  
  
**RUN:**  
\* All expressions are producing correct answers in your run.  Perfect.  
  
  
**SOURCE:**  
\* Style was good  
\* tabs removed  
\* indentation well done  
\* lines all <= 80 chars.  
  
  
**SOURCE LOGIC:**  
\* Main class correctly named Foothill.  
\* Very good logic, good type coercion.  
\* Good reuse of variables.  
  
  
As always, I have included the sample solution for option A.

 1 of 1

public class Foothill

{

public static void main(String[] args)

{

int myId, nLet, intResult;

double doubleResult;

String lastName;

myId = 11203484;

nLet = 10;

lastName = new String("My last name is Maloletkin \n");

System.out.println(lastName + "My student ID is " + myId + "\n"

+ "The number of characters in my last name is " + nLet + "\n");

intResult = myId % 2;

System.out.println("Expression #1 --------- : " + intResult + "\n");

intResult = myId % nLet;

System.out.println("Expression #2 --------- : " + intResult + "\n");

doubleResult = (double) myId / (nLet + 1000);

System.out.println("Expression #3 --------- : " + doubleResult + "\n");

intResult = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + nLet;

System.out.println("Expression #4 --------- : " + intResult + "\n");

doubleResult = (double) (10000 \* (nLet + 30) \* (nLet + 30))

/ (99 \* (nLet + 30) \* (nLet + 30) + (myId - 543210));

System.out.println("Expression #5 --------- : " + doubleResult);

}

}

/\*----------paste of run from console window------------

My last name is Maloletkin

My student ID is 11203484

The number of characters in my last name is 10

Expression #1 --------- : 0

Expression #2 --------- : 4

Expression #3 --------- : 11092.558415841584

Expression #4 --------- : 55

Expression #5 --------- : 1.4789243117964364

----------------------------------------------------- \*/