Preparing People to Lead Extraordinary Lives
LOYOLA UNIVERSITY CHICAGO
DEPARTMENT OF Computer Science

# Comp 170 Assignment – Advanced Programming Assignments

These assignments are intended to expand your thinking about programming. They are more open ended than the original course assignments. They ask you to do less structured tasks.

You should always start with a clear statement of what you intend to do. This "Problem Statement" is your summary of the program and your goals for how it will look and respond. Include sample input and output. Then write pseudo code (PIPO); then "real" code. I suggest you take these one step at a time, add one function and get it to work before you go onward to more.

Sometimes the best way to get a good program is to THROW OUT the one you have in front of you and start over. Remember Tony Hoare!

Note: Number 4 – 8 are adapted from Deitel & Deitel, *Visual C#* How to Program or *Java How to Program.*

## 0. Console IO

Well, console is a bit boring, but it can really do quite a bit. Learn how to control and process what the user types. Learn to be able to get multiple things on one line. Learn how to know when the line is done and when to ask for more input from the user.

Study all the Scanner methods that start with next….

Here's an example task to get started, then make up some more on your own.

Prompt the user to enter a single word, an integer, a floating point number, another integer, a number in scientific notation, then a sentence of zero to a large number of words. The input should be on one line (or more than one if you want to allow really long words and numbers). Separate the first words and numbers with a blank. End the sentence with a period. Put each item the user enters into a variable of the correct type (String, int, double)

# 1. Work with Dates

Text chapter 4, Programming Project 4 gets a valid date from the user. Expand this to do some arithmetic with dates. Once the user has provided a valid date, ask them for more input: a number of days, weeks, months, or years to add to that date. Calculate the new date and return it to them.

Can you make it work so that the user inputs any combination of days, weeks, months, years all on one line?

Don't forget: leap years!

Even more interesting: When you print out the date, say what day of the week it will be. How would you do that??? (Hint: there are only 14 possible yearly calendars, and 7 of those are for leap years).

If you want to keep trying things out, after you have started to do the above on your own, see how much Java can do for you:
https://docs.oracle.com/javase/tutorial/datetime/iso/temporal.html

# 2. Loop with Numbers

This problem is famous with many variations.

Ask the user for any integer number (positive or negative). Call this the target number.

Then, get numbers from the user and add each number entered to all the other numbers entered until the sum matches the target number exactly. When the target is reach, congratulate the user, print out how many numbers it took (numbers the user entered after the target), and end the program.

Options: give the user hints such as "too high" or "too low", better hints might be "getting close", "way off", etc. How about when they are within one of the target saying "Are you a programmer? You are off by one!". Maybe consider, "You are off by 10 or less", "Off by more than 100", etc.

The real game has the program pick the number using a random number. See Math.random(). You can pick the range or have the user specify (e.g. the target will be between -10,000 and +10,000). You can do this by adding up the numbers the user entered, or by just having them guess the random target.

## 3. Sieve of Eratosthenes

A prime number is any integer greater than 1 that's evenly divisible only by itself and 1. The Sieve of Eratosthenes is a method of finding prime numbers. It operates as follows: a) Create a primitive-type boolean array with all elements initialized to true. Array ele- ments with prime indices will remain true. All other array elements will eventually be set to false. b) Starting with array index 2, determine whether a given element is true. If so, loop through the remainder of the array and set to false every element whose index is a multiple of the index for the element with value true. Then continue the process with the next element with value true. For array index 2, all elements beyond element 2 in the array that have indices which are multiples of 2 (indices 4, 6, 8, 10, etc.) will be set to false; for array index 3, all elements beyond element 3 in the array that have indices which are multiples of 3 (indices 6, 9, 12, 15, etc.) will be set to false; and so on.

When this process completes, the array elements that are still true indicate that the index is a prime number. These indices can be displayed. Write an application that uses an array of 1,000 elements to determine and display the prime numbers between 2 and 999. Ignore array elements 0 and 1.

## 4. Calculating Pi

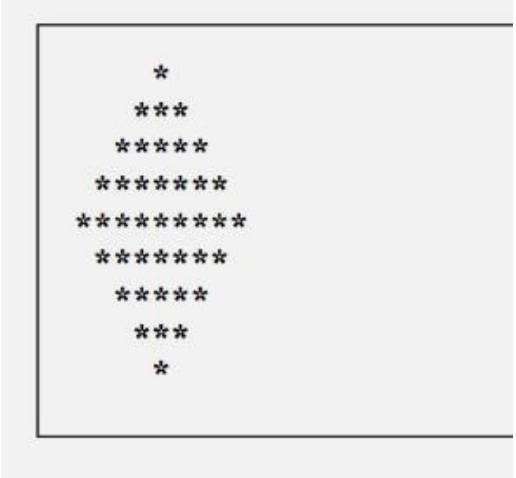
Calculate the value of π from the infinite series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

Display a table that shows the value of π approximated by computing one term of this series, by two terms, by three terms, and so on. How many terms of this series do you have to use before you first get 3.14? 3.141? 3.1415? 3.14159?

## 5. Display a Diamond

Write an app that displays the following diamond shape. You may use output statements that display a single asterisk (*), a single space or a single newline character. Maximize your use of repetition (with nested for statements) and minimize the number of output statements




(Modified Diamond Program) Modify the app you wrote to read an odd number in the range 1 to 19 to specify the number of rows in the diamond. Your app should then display a diamond of the appropriate size.

## 6. Dice Rolling

Write an app to simulate the rolling of two dice. The app should use an object of class Random once to roll the first die and again to roll the second die. The sum of the two values should then be calculated. Each die can show an integer value from 1 to 6, so the sum of the values will vary from 2 to 12, with 7 being the most frequent sum and 2 and 12 the least frequent sums. Your app should roll the dice 36,000 times. Use a one-dimensional array to tally the number of times each possible sum appears. Display the results in tabular format. Determine whether the totals are reasonable (e.g., there are six ways to roll a 7, so approximately one-sixth of the rolls should be 7).

…Continue on to various dice games as you wish….

## 7. Pig Latin

(two approaches, use regular expressions, use String split and arrays. For each loops are helpful too)

Write an app that encodes English-language phrases into pig Latin. Pig Latin is a form of coded language often used for amusement. Many variations exist in the methods used to form pig Latin phrases. For simplicity, use the following algorithm: To translate each English word into a pig Latin word, place the first letter of the English word at the end of the word and add the letters “ay.” Thus, the word “jump” becomes “umpjay,” the word “the” becomes “hetay” and the word “computer” becomes “omputercay.” Blanks between words remain blanks. Assume the following: The English phrase consists of words separated by blanks, there are no punctuation marks and all words have two or more letters.

Enable the user to input a sentence. Method GetPigLatin should translate a single word into pig Latin. Keep a running display of all the converted sentences in a text box.

## 8. Random Sentences

Write an app that uses random-number generation to create sentences. Use four arrays of strings, called article, noun, verb and preposition. Create a sentence by selecting a word at random from each array in the following order: article, noun, verb, preposition, article, noun. As each word is picked, concatenate it to the previous words in the sentence. The words should be separated by spaces. When the sentence is output, it should start with a capital letter and end with a period. The program should generate 10 sentences and output them. The arrays should be filled as follows: The article array should contain the articles "the", "a", "one", "some" and "any"; the noun array should contain the nouns "boy", "girl", "dog", "town" and "car"; the verb array should contain the past-tense verbs "drove", "jumped", "ran", "walked" and "skipped"; and the preposition array should contain the prepositions "to", "from", "over", "under" and "on".

Version 2.0 (Oct 2017)
Adding additional assignments, especially with loops
Version 1.0 (original)
AdvancedProgrammingAssignments
©W L Honig 2016 for Comp 170 Loyola University Chicago