ICS4M0 – Final Culminating Problems

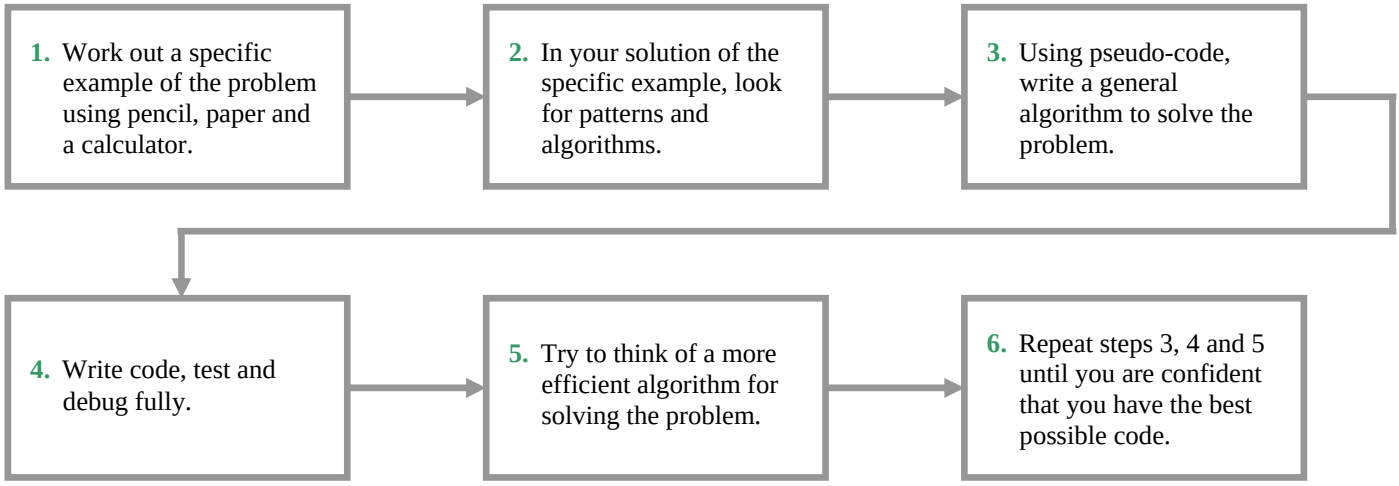
Introduction

The main purpose of this set of problems is to

- Help you remember the importance of analyzing problems and planning solutions BEFORE ATTEMPTING TO WRITE PROGRAMS TO SOLVE THE PROBLEMS! Keep in mind that a program is a set of instructions that a computer follows to solve a problem. If you do not know how to solve a particular problem, it is impossible for you to write a program that solves the problem!
- Consolidate your knowledge of programming concepts and programming skills.
- Help you to develop the facility for quickly choosing an appropriate programming structure.
- Build a library of frequently used code.

For instance, you will learn to decide quickly whether to choose a **for** loop, a **while** loop or a **do … while** loop. Furthermore, many of the problems presented below occur frequently while developing solutions to larger problems. For example, the problem of generating random numbers without repetition is very often encountered in many different programming situations.

When working on these problems, keep in mind the following problem solving model:



A Summary of Important Concepts required for Solutions of the Final Culminating Problems

- Arrays
- Selection Structures
- Counted and Conditional Repetition Structures (**for**, **while**, **do … while**)
- String Manipulation Methods (See "Using Strings in Java" in Unit 0)
- The Modulus Operator (%)
- Generate Random Integers
- Search an Array for a Given Value
- Sort Numbers from Smallest to Largest
- Sort Strings from Smallest to Largest
- The Binary Representation of Positive Integers
- The Twos Complement Binary Representation of Negative Integers
- Recursion

Mathematical Problems

1. The ancient Greek civilization had a keen interest in philosophy, mathematics, science, literature and the pursuit of knowledge. In fact, the Greeks were more interested in intellectual inquiry for its own sake than in practical applications because they believed that nature existed primarily for the wonderment of humans. Nature was there to be explored, contemplated and even worshipped, but not to be tampered with or altered. Largely due to this attitude, in a few short centuries the Greeks developed a body of knowledge and a system of rational thought that was unrivalled in ancient times. In fact, it was the rediscovery of ancient Greek learning that spawned the Renaissance, a pivotal period without which our modern technological society probably would not exist.

Among other things, the Greeks were interested in the properties of numbers, including numbers that the Greeks called ***perfect***. An integer is called ***perfect*** if the sum of its ***proper divisors*** is equal to the number itself. Two examples of perfect numbers are 6 and 28 because 6 = 1 + 2 + 3 and 28 = 1 + 2 + 4 + 7 + 14.

(a) Write a Java method that can determine whether a given number is perfect.
(b) Write a Java method that finds ***all*** perfect numbers within the range of a Java **short** variable.
(c) The numbers 220 and 284 are called an ***amicable pair*** because the sum of the proper divisors of 220 is 284 and the sum of the proper divisors of 284 is 220. Write a Java method that determines whether a given pair of integers forms an amicable pair.
(d) Write a Java method that finds ***all*** amicable pairs within the range of a Java **short** variable.

2. To a mathematician, a prime number serves the same purpose as a chemical element does to a chemist. Just as all molecules are made using the elements of the periodic table, all numbers can be constructed using nothing but primes. Another way of putting this is that the primes are the basic building blocks of all numbers. Furthermore, prime numbers are not just a whimsical curiosity of mathematicians. Without them, it would not be possible to conduct secure transactions over the Internet! (For more information on how prime numbers help to secure Web transactions, consult http://en.wikipedia.org/wiki/Public_key_encryption)

(a) What is a prime number?
(b) Rewrite your definition in 2(a) using the concept of ***proper divisor*** (see question 1).
(c) Explain how you could use your code for finding all proper divisors of a number to determine whether a given number is prime. Would this be an efficient method?
(d) Write a Java method that can determine whether a given number is prime. (When you test your program, use relatively small integers as input. Otherwise, you may spend a great deal of time waiting for your program to produce its output.)
(e) Primes are considered the building blocks of all numbers because every integer can be written as a product of primes. For example, 24 = 2(2)(2)(3) = 2^3(3) and 105 = 3(5)(7). Write a Java method that can find the prime factorization of a given integer. (Don't forget to observe the ***caveat*** of 2(d).)

3. Write a Java method that will convert any Java **int** value specified in ***decimal*** (base ten) form to ***binary*** (base 2) form. Note that to convert ***negative*** integers to binary form, you must understand the "twos complement" binary representation of negative integers (see unit 2).

4. Write a Java method that can factor any simple trinomial (i.e. a trinomial of the form x^2+bx+c) that is factorable over the integers. For example, the simple trinomial $x^2-15x+56$ factors over the integers as $(x-7)(x-8)$. On the other hand, x^2-5x+1 does not factor over the integers. That is, it is impossible to write x^2-5x+1 in the form $(x-n)(x-m)$ where *n* and *m* represent integers. To determine whether a simple trinomial factors over the integers, simply calculate its ***discriminant***, b^2-4ac.

If b^2-4ac is a ***perfect square***, then the simple trinomial ***factors*** over the integers.

If b^2-4ac is ***not a perfect square***, then the simple trinomial ***does not factor*** over the integers.

e.g. $x^2-15x+56$: $b^2-4ac=(-15)^2-4(1)(56)=225-216=9$

9 ***is*** a perfect square because it is the square of an integer, that is $9=3^2$

e.g. x^2-5x+1: $b^2-4ac=(-5)^2-4(1)(1)=25-4=21$

21 ***is not*** a perfect square because it cannot be written in the form n^2, where *n* is an integer.

5. Euclid's method for calculating the greatest common divisor of two integers is based on the following relationship:

$$\gcd(m, n) = \gcd(n, m \bmod n)$$

The example below shows how this property can be applied repeatedly to calculate the gcd of two integers:

gcd(2322, 654) = gcd(654, 2322 mod 654) = gcd(654, 360)
gcd(654, 360) = gcd(360, 654 mod 360) = gcd(360, 294)
gcd(360, 294) = gcd(294, 360 mod 294) = gcd(294, 66)
gcd(294, 66) = gcd(66, 294 mod 66) = gcd(66, 30)
gcd(66, 30) = gcd(30, 66 mod 30) = gcd(30, 6)
gcd(30, 6) = gcd(6, 30 mod 6) = gcd(6, 0)
gcd(6, 0) = 6

$\therefore$ gcd(2322, 654) = gcd(6, 0) = 6

The Java method shown at the right is an ***iterative implementation*** of Euclid's method for calculating the greatest common divisor of two integers *m* and *n*.

Write a ***recursive implementation*** of Euclid's algorithm.

```
public static int gcd(int m, int n)
{
    while(n!=0){
        int temp=n;
        n=m%n;
        m=temp;
    }
    return m;
}
```

Array and String Problems

1. Write a Java method that fills an array ***at random*** with integers in the range specified by the parameters `lowest` and `highest`. Your method should be able to process whatever array is passed to it.

2. Write a Java method that returns a string consisting of all possible ways to choose a pair of numbers from a list of integers ranging from 1 to *n*. The value of *n* is specified by the parameter *n*. Note that order does not matter in this problem. For example, the pair "2, 5" is considered the same as the pair "5, 2."

3. Write a Java method that removes any repeated numbers from an array without leaving any gaps.

4. Write a Java method that determines the ***mean*** (average) of an array of numbers.

5. We have learned (or shall learn very shortly) a fast method of sorting data called quicksort. Although quicksort is very fast, it involves recursion, which is difficult for some students to understand. A method known as "bubble sort" is an alternative that most students find much easier to implement. On average, the "bubble sort" algorithm is much slower than quicksort. However, for small sets of data, the difference in the execution speeds of the two algorithms is imperceptibly small.
   - **(a)** Use the Internet to learn about the bubble sort algorithm.
   - **(b)** Write a Java method that uses bubble sort to sort an array of numbers in ***ascending*** order (from lowest to highest).
   - **(c)** Write a Java method that uses bubble sort to sort an array of numbers in ***descending*** order (from highest to lowest).
   - **(d)** Write a Java method that uses bubble sort to sort an array of numbers in either descending or ascending order. The order should be specified by a parameter. (A **boolean** parameter should work well.)

6. Write a Java method that determines which number(s) occur most frequently in an array of numbers (i.e. find the ***mode***). **Hint:** First sort the array using one of the methods that you developed for question 5.

7. Write a Java method that determines the ***median*** of an array of numbers. That is, find the number that divides the array into two halves, one half consisting of all the numbers ***less than*** the median and the other half consisting of all the numbers ***greater than*** the median.

8. Write a Java method that removes ***all*** the vowels from a word.

9. Write a Java method that removes ***all*** the spaces from a string.

10. Write a Java ***recursive method*** that takes an integer and returns it as a string with commas inserted in the correct places. For example, if the integer 19866767 were passed to the method, the string "19,866,767" would be returned.

11. Determine whether a given string is a ***palindrome***. (A ***palindrome*** is a word or phrase that reads the same forward or backward. Examples of palindromes include "bob," "madam" and "ten animals I slam in a net.")
Find ***both*** an ***iterative*** and a ***recursive*** solution to this problem. **Hint:** Use your method from question 9 to remove all spaces from the given string before checking whether it is a palindrome.

12. Write a Java method that scrambles the letters in a word.

13. Write a Java method that conjugates any regular French verb. For ease of implementation, the correctly conjugated verb should be returned as a single string. If you need to brush up on your French grammar, here are the correct endings for the three types of regular verb infinitives:

-er verbs				-re verbs				-ir verbs			
je	-e	nous	-ons	je	-s	nous	-ons	je	-is	nous	-issons
tu	-es	vous	-ez	tu	-s	vous	-ez	tu	-is	vous	-issez
il	-e	ils	-ent	il	-	ils	-ent	il	-it	ils	-issent
elle	-e	elles	-ent	elle	-	elles	-ent	elle	-it	elles	-issent
on	-e			on	-			on	-it		

For example, if the infinitive "vendre" were passed to the method, it would return the following string:
"je vends, tu vends, il/elle/on vend, nous vendons, vous vendez, ils/elles vendent"

14. Write a Java method that randomly generates a set of six integers ranging from 1 to 49 (for Lotto 6/49). The numbers should be displayed in ascending order (smallest to largest).