# Homework 1

Complete, to run on our csci2 Unix server, the following Programming Challenges from Gaddis:

I. 3.19
II. 3.20
III. 3.21
   - This problem modifies problem 3.20.
   - Use the statement {**`const double PI = atan(1.0)*4.0;`**} to compute the value of π.

IV. 3.24
   - Write your input statements so that user responses can include spaces (*multi-word*).
   - This type of problem is an example of a Mad-Lib.

V. Use the two division operators to write a program to calculate the minimum number of U.S. coins necessary to make the input value, in the range $0 - $1, and how many of each.
For example, given an input of $0.64, your program should print:

```
2 quarters
1 dimes
0 nickels
4 pennies
------------
7 coins total
```

For each programming challenge, at least one day prior to submitting your source file (as a syntax-highlighted PDF file) and output to the D2L dropbox, you will submit a pseudocode (*see the top of* Gaddis p. 134) text file for that problem to the D2L dropbox.

Each Programming Challenge problem shall be solved on our csci2 Unix server by a coupled pair of source and executable files.

Each source file will include the following comments:

- Your Name
- "CSCI 201"
- Your section number
- The current semester
- "Homework 1"
- The name of the program and the Programming Challenge number it is solving
- The full pathname on our csci2 Unix server for its executable file.
- A brief description of the purpose of the program

For each source file you will:

- Import the pseudocode statements into your editor and transform the pseudocode statements into internal comments to describe what your code is doing and why. You shall then write your C++ statements below each comment derived from the pseudocode.
- Use descriptive identifiers
- Use both vertical and horizontal white space consistently to enhance readability.
- Use comments which describe your variables while defining in a style as demonstrated in Program 3-28, lines 15 – 22.

**Testing**: test with several input values. Validate your program, when possible, by repeating the calculations, using the input values, on a scientific calculator.

Use the UNIX `script` command to generate an output file (.txt) consisting of several runs of your source file's executable with different test (input) data. One of those runs should replicate the author's example if stated.