**CSC 4992**
**Section 002**
**Special Topics in Computer Science**
**Python Programming**
**Winter Term 2018**
**Assignment 01**
**40 points**
**Due 02/8/2018 (11:59 P.M.)**

Submit a source file (.py) for each question. Put them all in one folder, compress then post to blackboard.

**Problem 1 (10 points)**
Compute a person's income tax.
1.1 Significant constants
    tax rate
    standard deduction
    deduction per dependent
1.2. The inputs are
    gross income
    number of dependents
1.3. Computations:
    taxable income = gross income - the standard deduction - a deduction for each dependent
    income tax = is a fixed percentage of the taxable income
1.4. The outputs are the income tax

**Problem 2 (10 points)**
Write a program that that takes two numbers as input and returns the product, the maximum and minimum of them.

**Problem 3 (30 points)**
**Login security**

One important aspect of security in computer science is the concept of *hashing*: taking some text, and somehow converting it to a number. This is needed because many security algorithms work through math, so numbers are needed.

Another important aspect is the use of the modulo operator (%). You've seen this -- it returns the remainder portion of a division. This is useful because unlike most other math operators, modulo is one-way. That is, I can tell you that I'm thinking of a number *x*, and when I mod it by 5, I get 3, but from this information alone, you don't know whether *x* is 3 or 8 or 13 or 18, or ...

In this problem, we'll create a login screen, where the user must enter a password in order to see a secret message. We will give the user **3 chances** to get the password right, and either print the secret message or a failure message (after 3 chances).

First, write a program that takes one string. It will hash the string using the built-in Python function **hash** (try it on the shell) and modulo the value by a prime number (e.g. 541 -- this is very small in the computer science world but good enough for us). The program should then print this number.

e.g. It should print **211** (result) when the string is **"mypassword"** (if you use 541 as the prime, for example) At the top of the file, define a variable **_KEY** to be the result.

- Now, write the rest of the program. Each time you ask the user for the password, ask for the user's input, calculate the key and compare the value to **_KEY**. If the two match, the user (most likely) entered the correct password, otherwise he loses one chance.