CPSC 110 **Lab 01** Due: Fri Sept. 13, 2024, 9am PST

### General Lab Instructions:

1. All assignments will be submitted through Canvas. If you submit multiple times, the most recent submission will be graded.
2. You will typically explain your code from the last lab to the instructor at the beginning of the next lab to receive your full marks and feedback.
3. If you work with other people, list all of your names on the assignment. Each assignment will specify how many others you can work with.
4. Please review the collage's plagiarism policies. As a general rule, copy/paste should never be used for any code. Ever. You may discuss topics with other students and reference online materials, but the final assignment must be written by the student. If you reference materials/people besides the textbook/instructor then cite this in a file `contributions.txt`. Full citations are not necessary, a bullet point list of the materials/people that you references is suitable.
5. Code will be graded for functionality as well as quality. This means code that should be easy to read with descriptive variable names and appropriate comments.
6. You should aim to stick to the concepts we have already covered in class. For example, do not use "if statements" on assignment 1 since it was not covered in lecture 1. If you have studied programming elsewhere then it is tempting to jump ahead, but full marks will only be given to assignments that are focused on this course's content.
7. Code that cannot run successively in Python 3 will receive a zero. Make sure you are using the correct python version and allow yourself enough time for thorough testing.
8. At the start of each assignment will be a list of the required files for submission. All files should be compressed in one zip folder named `<studentnumber>`$_{<yourname>\_LAB<number>\_CPSC110.zip}$. For example: `21870110`$_{L}UKESKYWALKER\_LAB01\_CPSC110.zip$

## Assignment 1 Instructions:

- Total Points: 30
- Due: Fri Sept. 13, 2024, 9am PST
- Group size: You may work with one other person
- Required files:
    1. `arithmetic.py`
    2. `errors.txt`
    3. `easter.py`
    4. `checkout.py`
    5. (Optional) `contributions.txt`

# 1 Getting Started [5 points]

Ask the user to enter a positive 4 digit number. Then print:

- The sum of all the digits of the number
- The product of all the digits of the number
- The number backwards
- The number doubled
- The square root of the number

Save the program in a file `arithmetic.py`

# 2 Debugging Challenge [10 points]

Unlike other holidays such as Christmas and Remembrance Day, Easter does not have a fixed date in our calendars. Instead it is celebrated on the Sunday after the first full moon following the spring equinox. This can occur between March 22 and April 25. The date of Easter can be calculated using the Anonymous Gregorian Computus algorithm.

Below you are given the algorithm to calculate the date of Easter. You must implement this algorithm and keep track of the bugs that you fix along the way. Your program should accept a year from the user and then print the date in the format "DD MM YYYY"

a) Read the algorithm carefully and make sure you understand the instructions. Next you will need to write this algorithm in Python.

   As you debug your code, keep track of the errors you must fix. There is no wrong answer to this, you may have 10 errors to fix 100 errors to fix. Each time you fix an error, write down:

   1. The error type (ie syntax, runtime or semantic error)
   2. The line the error occurred (ie line 10)
   3. How you fixed the error (ie "2f was changed to 2*f")

   Put all this in a file called `errors.txt`.

b) Now that you have written the code, save your completed Python program in a file `easter.py`.

The algorithm is shown below:

- Set $a$ equal to the floor of $year$ divided by 100
- Set $b$ equal to the remainder when $year$ is divided by 100
- Set $c$ equal to the floor of $b$ divided by 4
- Set $d$ equal to the remainder when $b$ is divided by 4
- Set $e$ equal to the remainder when $year$ is divided by 19
- Set $f$ equal to the remainder when $a$ is divided by 4
- Set $g$ equal to the floor of $a$ divided by 4
- Set $h$ equal to the floor of $\frac{a+8}{25}$
- Set $i$ equal to the floor of $\frac{a-h+1}{3}$
- Set $j$ equal to the remainder when $19e + a - g - i + 15$ is divided by 30
- Set $k$ equal to the remainder when $32 + 2f + 2c - j - d$ is divided by 7
- Set $l$ equal to the floor of $\frac{e+11j+22k}{451}$
- Set $month$ equal to the floor of $\frac{j+k-7l+114}{31}$
- Set $day$ equal to one plus the remainder when $j + k - 7l + 114$ is divided by 31

# 3 Cash Register [15 points]

You have been asked to build the software of a self-checkout for a local bookstore. Since this is a small business, they only take cash and do not accept credit cards.

The program must do the following:

- Prompt the user to enter the dollar value of their purchase
- Print the sales tax (5% GST and 7% PST) and the total value owed
- Promet the user to enter the cash paid
- Print the total change owed and the coins given

The system will distribute change with the following currency:

| **Coin Name** | **Value** |
|---|---|
| Penny | $0.01 |
| Nickel | $0.05 |
| Dime | $0.1 |
| Quarter | $0.25 |
| Loonie | $1 |
| Toonie | $2 |
| Fin | $5 |
| Tenner | $10 |

Some extra notes:

- It helps to remember the Units of Time exercise from the worksheet.
- Your program can assume that the user will always enter a valid price. If the user enters something that isn't the price, your program will terminate and print an error. That's okay, it's not your job to handle it. (yet)
- If the user tries to give you less money than is owed, the change will be incorrect. That's okay, it's not your job to handle it. (yet)
- You may get numbers that look odd. In python 5.98 - 2.68 = 3.3000000000000003, this is not your error. We will explain why in CPSC 121, but for this class you should ignore it. If it really bothers you, you can do some extra reading on the topic here: https://docs.python.org/3/tutorial/floatingpoint.html

Write a program in Python to meet the specifications outlined above. Save the program in a file `checkout.py`

An example output is below (black text is output from the program, red text was typed by the user):

```
Please Enter the Price of Your Purchase:  55.25
GST: $ 2.7625
PST: $ 3.8675
-----------------
Total:  $ 61.88
Enter Cash Paid:  80
18.119999999999997 is equal to:
1 Tenners
1 Fins
1 Toonies
1 Loonies
0 Quarters
1 Dimes
0 Nickles
1 Pennies
```