Introduction to Computer Science (ICS 3U)
Three Structures in Programming Culminating Activity

Instructions:

1) The challenge problem at the end and the extensions to the other problems are to be done only when **all** the required programs are created.

2) Make sure that your programs properly prompt the user for input and correctly identify all output.

3) Make sure to document all problems by including your name and a description of the input and outputs program. Make sure you use descriptive variable names.

4) Save all of your problems in a folder called **Structures Culminating Assignment**.

5) I encourage you to use web sites as resources to help you learn but any code that you use in your programs that we have not discussed will be discounted unless **you** can explain how it works how it works in your documentation and orally to the teacher.

6) While I also encourage you to help each other, every person must submit his or her own set of programs. Copying another person's program is PLAGERISM and will be dealt with accordingly.

1. Create a program which will allow the user to input two integers. If the 2nd integer is larger than the 1st, the program should then output all of the integers between the two inputted integers. (ie. if the user inputted 3 and 8, the computer would output 3, 4, 5, 6, 7, 8). If the 2nd integer is smaller than the 1st. The program should then output all of the integers between the two inputted integers in reverse order. (ie. if the user inputted 8 and 3, the computer would output 8, 7, 6, 5, 4, 3). If the two integers are the same, output an appropriate message. Call your program **MajorOne1.py**

   Hint: You will have to use repetition structures within a selective structure. These are called nested structures**.**

2. The following item appeared in the Readers Digest:

   > "***Magic Reasoning.*** Take your house number and double it. Add five. Multiply by half a hundred, then add your age (if you are under 100). Add the number of days in the year, and subtract 615. The last two figures will be your age; the others will be your house number."

   Create a computer program which will allow any user to see if this "magic reasoning" works for them. Call your program **MajorOne2.py**

   Extension: Create a user-defined function to perform the "magic reasoning" calculation for them.

3. Create a program that will simulate a series of races between two (or more) "horses". The program will first ask the user for the name of each horse. It will then ask the user how many races they wish to run. For each race, the program should output the race time of each horse in seconds. The program should then determine which horse won or if there was a tie and output an appropriate message. After all the races have been run, the program should output the name and the total number of wins, losses and ties for each horse. Call your program **MajorOne3.py**

   Hint: Use the random integer generator to generate the race time of each horse.

   Extension 1: Make the time outputted to the nearest 100^{th} of a second.

   Extension 2: Actually draw two or more horses (or a shape representing horses) "race" across the screen. Note that their movements must correspond with their time.

4. Write a program to play the game rock, paper, scissors. Each player (you and the computer) chooses one of these 3 things: rock, paper, or scissors. The player that wins depends on what the choice are:
   - rock beats scissors
   - scissors beats paper
   - paper beats rock

   The computer will make its choice randomly. Display the choice of the computer so the user can see it, and display an appropriate message once you've calculated who won. Keep playing until the user decides to quit. Keep track of how many times the user wins, how many times the computer wins, and how many ties there are, and display this information at the end. Call your program **MajorOne4.py**

   Extension: Modify your program so that the computer makes a better choice than a random one.

5. The government of the United Federation of Murphville levies income tax on its citizens in stages based on their taxable income. Create a program that allow the user to input their full name, address, city, postal code, social insurance number and their income. Their income tax should be calculated to the nearest cent. The table used by the government is:

   - There is no tax paid if the income is less than the minimum taxable salary of $20 000.
   - If a person makes more than the minimum taxable salary, the tax rate on the first $ 50 000 is 25%.
   - On the next $50 000, the tax rate is 35%.
   - The tax rate is 50% on the remainder.
   - There is also a surtax (additional tax) of $1 000 charged to those who make more than $120 000.

   All of the data for each user should be outputted in an attractive manner. The program should also total all the taxes paid by all the Murphville citizens inputted. The program should work for any number of users. Call your program **MajorOne5.py**

   Extension: Modify your program so that the data is inputted from a file called **Citizen.txt**.

Challenge Problem

6. Create a program which simulates choosing 5 cards from a deck to create a poker hand. The cards should be displayed as "the king of hearts" or "the three of clubs". This program must use the random integer generator. Call your program **MajorOne6.py**

   Hint: Use the random integer generator to generate the suit and another random integer generator to generate the card.

   Extension 1: Make sure that no cards are repeated in a player's hand (ie. there should not be two "aces of clubs"). **Hint**: Use lists.

   Extension 2: Tell the user how good of a hand he/she has. For example, you could tell the user he has a pair, a full house, three of a kind, a straight, a royal flush, etc. See me or a poker playing friend or parent if you are not sure about poker hands.