FCS Programming Warm-Up (Scratch)

Before we learn new concepts and dive into a new programming language we will review a few important principles we have already covered before.

Use Scratch at https://scratch.mit.edu/ to program the following.

For each of the following programs, you should follow the steps below:

1. Make sure you understand the problem in terms of
    a. what are the inputs?
    b. what "processing" do you need to do?
    c. what is the output you need to produce/display?
2. Write the algorithm in pseudocode.
3. program your solution
4. Run it and make sure you test it with different inputs and get correct outputs.

Program 1 - Mad Libs

This will review **variables** and **conditionals**.

Write a program which asks the user for 1 noun, 1 verb, 1 adverb, and 1 adjective. Then it generates a sentence/story using these words.

For example, if the user provides the words "dog", "jump", "fantastically", and "hilarious", you could produce something like: Did you and your hilarious dog jump on the bed? That's fantastically dangerous!

Extra Experience: ask the user if they want to enter more than 1 noun and 1 verb, and if they do, create a sentence/story that is different from the one you would create if they provide only 1 noun and 1 verb.

Program 2 - Simple Math

This will review **operations** (operators) and **custom blocks** (functions).

Write a program which asks the user for 2 numbers and then calculates and displays their sum (addition), difference (subtraction), product (multiplication), and quotient (division).

1. The program should only accept positive numbers, and keep asking the user for them, until they provide them.

2. Each **math operation** should be done using a **custom block** (for a total of 4 blocks), which will receive as input parameters the two numbers provided by the user.
    a. For a **refresher/reminder of custom blocks** look up the Scratch Concepts and Principles doc.

Extra Experience:
1. The program should restrict the input numbers to just integers (whole numbers), and keep “nagging” the user until it gets them.
2. Ask the user if they want to also see the remainder (modulo; left-over) of dividing the bigger number by the smaller number, and if the answer is ‘yes’, you should display this number as well.

Program 3 - Password Strength

This will review **operations** (operators), and **loops**.

Write a program which asks the user for a password and let’s them know if it is strong or not, following these criteria:
- if the password contains less than 8 characters, reject it, and ask for a longer password.
- if the password starts with 2 digits (numbers between 0 and 9), tell the user that their password is **very weak**.
    - E.g., 12abcdefg, 345678910
- if the password starts with 1 digit followed by a letter, followed by a digit, tell the user that their password is **adequate**.
    - E.g., 1a2bcdefg, 3x4y5zabc

Hints:
- Use the “length of” block, and the “letter 1 of word” block
- To check if a single letter/character is a number you can check if it is bigger that -1 and smaller than 10

Extra Experience:
1. enhance your program to also check if the user password contains at least 4 letters and 4 digits, and if so, tell the user that their password is **strong**.

Program 4 - Leap Year?

Ask the user for a year, and implement the algorithm for determining if a year is a leap year following the steps below:

1. If the year is evenly divisible by 4, go to step 2. ...
2. If the year is evenly divisible by 100, go to step 3. ...
3. If the year is evenly divisible by 400, go to step 4. ...
4. The year is a leap year (it has 366 days). Stop.
5. The year is not a leap year (it has 365 days).

E.g. leap years: 1200, 2000