Activity: Loops

Before You Begin:

Go thoroughly read the SDI_LogicLoops.pdf! This contains the full instructions for this project and the rubric, which explains how you will be graded.

Criteria:

For each problem below you will need the following:

- Label the section of code appropriately
- Prompt the user for each variable that is in the "User Input" Section of that problem.
- Validate each user prompt with a **while loop** to insure that the user is typing in a valid response. Remember we check text string different than numbers!
- Convert each user response to the **correct** data type, if needed.
- The result should be calculated using variables, not literal values when possible.
- Create code that will make decisions based on the value of the user's responses.
- Print the result of the decision-making to the console using the format given in the "Results" section of that problem.
- After each section put in a multi-lined code for the Test Values.
- Use only code and techniques learned in this class.

Graded Problems:

Problem #1 – For Loop: Piggy Bank

In this problem you will be asking the user how much money is in their piggy bank to start with. You will then ask them for an amount they will add each month. You will then tell them the total they will have after 1 year.

Prompt the user for the starting value of their piggy bank, make sure to validate this number and convert it to the correct data type for money.

Then prompt the user for an amount that they will put in every month for the next yea, make sure to validate this number and convert it to the correct data type for money.

Next create a for loop that will cycle one time for each month in a year. Each month add in the new amount to the piggy bank and tell the user how much is now in their bank.

- **User Inputs:**
  - Starting amount in the piggy bank
  - Amount added to the bank each month

- **Result To Print Out:**
  - "This month you now have **SX** in your piggy bank!"

- **Data Sets To Test**
  - Test 1:
    - Initial Amount – "Ten" – re-prompt because that is not a number - $10.00
    - Amount to add each month – $5.50
    - Output
      - "This month you now have **$15.50** in your piggy bank!"
      - "This month you now have **$21.00** in your piggy bank!"
      - "This month you now have **$26.50** in your piggy bank!"
      - "This month you now have **$32.00** in your piggy bank!"
      - "This month you now have **$37.50** in your piggy bank!"
      - "This month you now have **$43.00** in your piggy bank!"
      - "This month you now have **$48.50** in your piggy bank!"
      - "This month you now have **$54.00** in your piggy bank!"
      - "This month you now have **$59.50** in your piggy bank!"
      - "This month you now have **$65.00** in your piggy bank!"
      - "This month you now have **$70.50** in your piggy bank!"
      - "This month you now have **$76.00** in your piggy bank!"

  - Test One Of Your Own & Write both of these tests in a multi-lined comment in your code.

Problem #2 – For Loop: 3, 2, 1 Blast Off!

In this problem you will be making a countdown clock for a rocket launch.

Explain to the user the premise of the program, then prompt the user for the starting number to countdown from. Make sure to use validation to make sure this is a whole number and does not contain a decimal.

Next create a loop that will output a number on its own line. Start with the number that the user gave and each time it loops subtract one from this value. Stop the loop when you reach zero. After the loop, output to the user "Blast off!".

- **User Inputs:**
  - Number to start the countdown from
- **Result To Print Out:**
  - Output the numbers starting with the user prompted number, subtract one each time until you get to zero.
- **Data Sets To Test:**
  - Starting Number – **5**
    - Output
      - 5
      - 4
      - 3
      - 2
      - 1
      - 0
      - Blast Off!
  - Starting Number – "**Seven" –** This is not valid, so re-prompt the user -> **7**
    - Output
      - 7
      - 6
      - 5
      - 4
      - 3
      - 2
      - 1
      - 0
      - Blast Off!
  - Test One Of Your Own & Write all 3 of these tests in a multi-lined comment in your code.

Problem #3 – While Loop: Darn Good Donuts

The scenario for this program is that there is a box of donuts in the break room where the user works. The user will take this box and offer his co-workers donuts out of the box until he runs out.

Start by prompting the user for the total number of donuts that are in the box, making sure to validate that this number is greater than zero.

Create a loop that will keep going as long as there are donuts still in the box.
Inside of this loop, prompt the user for the amount of donuts that they would like to eat. To make sure there are no food hogs, each person can only have up to 3 donuts. So validate that the user is typing in a value from 0 to 3 donuts. Zero must be a valid option, because not everybody will want one.

Continue prompting people for donuts until the box is empty, you might end up with a negative value of donuts (Say there are 2 left in the box, but the user wants 3). You will have to test for this after the loop.
If there are zero donuts left, tell the user "Donuts are all gone, hope everyone enjoys them!"
If there is a negative value left, tell the user that you will owe them those donuts tomorrow. Make sure this is a positive number, you can't owe someone -2 donuts.

- **User Inputs:**
  - Initial amount of donuts in the box
  - Inside of the loop, keep asking the user how many donuts they would like to eat, 0 –> 3 are the only valid choices.
- **Result To Print Out:**
  - At the start of the loop, you should tell the user how many donuts are left in the box.
    - "There are **X** donut(s) left in the box."
      - Where **X** is the donuts still left.
  - After the user gives out all of the donuts…
    - If there are zero left
      - "Donuts are all gone, hope everyone enjoys them!"
    - If there is a negative number left
      - "I ran out of donuts, I will have to owe you **Y** donuts tomorrow."
        - Where **Y** is a positive number of the donuts that you are missing.

- **Data Sets To Test:**
  - Test 1: Starting donuts in the box – 12
    - Co-worker requests – 2 donuts
      - "There are **10** donut(s) left in the box."
    - Co-worker requests – 0 donuts
      - "There are **10** donut(s) left in the box."
    - Co-worker requests – "three" donuts – re-prompt for a number - 3
      - "There are **7** donut(s) left in the box."
    - Co-worker requests – 5 donuts – this number is too high, re-prompt - 3
      - "There are **4** donut(s) left in the box."

        - Co-worker requests – 2 donuts
            - "There are **2** donut(s) left in the box."
        - Co-worker requests – 2 donuts
            - "Donuts are all gone, hope everyone enjoys them!"

- Test 2: Starting donuts in the box – 4
    - Co-worker requests – 3 donuts
        - "There are **1** donut(s) left in the box."
    - Co-worker requests – 0 donuts
        - "There are **1** donut(s) left in the box."
    - Co-worker requests – 2 donuts
        - "I ran out of donuts, I will have to owe you **1** donuts tomorrow."

- Test One Of Your Own & Write all 3 of these results in your comment section.