Control Flow

Complete this worksheet by answering the questions using the given program. Make sure to color or **bold** your answers so they are distinguishable from the question.

Getting Started

- ☑ ~~Make a copy of this document by selecting "Make a Copy" from the File menu.~~
- ☑ ~~Go to https://replit.com/~ and log in with Gmail using your Weber State email address.~~
- ☑ ~~Open the code linked below~~
- ☑ ~~Click **fork**, located in the upper right-hand corner, to create your own copy~~
- ☑ ~~In your copy, go to the upper right-hand corner, and click **invite**. Then, click the button **Generate a Join Link**.~~
- ☑ ~~Paste the generated link in the second bullet point below~~

Worksheet Program Links

- *Starter Code*: Repl Control Flow
  - *Finished Program Code*:https://replit.com/join/cskdvrgrdk-trinanixon1

Example 1

1. Observe the loops in example 1. The first one has Expected values of 0 1 2 3 4 5. Run the program. Does the loop print out the expected results? Yes.

2. Notice that the loop is duplicated for the rest of the questions. Your job is to modify the for loops, so it prints out the expected values.

3. Look in the main() method at the bottom of the page. Notice that example2() and example3() are commented out. Comment out Example1() and uncomment Example2(), so you can continue on to the next section.

Example 2

This example demonstrates the use of a loop to keep the program repeating until the user chooses to quit. The program will continuously allow the user to enter a number from 1 to 100 until the user chooses to quit by entering the value -1. It then reports the low and the high values entered.

1. Observe the condition of the do/while loop. What variable is used? num is the variable used in the condition of the do/while loop.

2. Run the program. Enter the value 5 when prompted. Did it show the value 5 as the highest number? Yes.

3. Now enter the value -1. What happened? The program quit.

4. Look at the condition for the first If Statement. When do you think this condition would evaluate to be true? When the user enters a -1 to quit the program.

5. Why do you think it did not show the value -1 as the low? Which statement makes it fall out of the loop? Because the break statement is what makes it fall out of the loop, so it never gets to the if statement 'low=num' where it assigns the low number.

6. Run the program again, this time enter 200. Did it display that 200 was an invalid number? Or, did it count the value 200 as the high? Yes, it displayed that 200 was an invalid number, and It counted 200 as the high.

7. The difference between the break and continue statement is that break jumps you out of the loop. Continue jumps you out of the current iteration of the loop and continues with the next iteration.

   Which do you think we should use (break or continue) on the invalid number in the if statement to make it skip the rest of the loop and jump to the top to allow them to enter a number again? Add the command into the if statement. Continue

8. Run the program again. Enter 200 for the value. Does 200 get set to the high? Does the program still tell you to enter a number? Then you did add the correct command? If not, try the other command. No, does not get set to the high. Yes, program says to enter a number after stating 200 was an invalid number.

9. Run the program again, use the numbers 5, 7, and 9. Does the high value keep changing? Yes .

10. Add the value 6. Did the high change this time? No.

11. Look at the variable declarations for the `high` and `low`. What value are they set to to begin? Both set to 0.

    Explanation: When you enter a value, the number you enter is compared against the high. If it is larger than the number, then it sets the high to that number. So, when you entered the value 5, since 5 was greater than 0, 5 became the high.

12. The low variable has similar logic, but there is an issue with the initialization. It is set to 0. When you enter the value 5, since 5 is not smaller than 0, then the low does not get set. What could you initialize the low variable to so that any number the user types in would be lower than the initial value? *(Hint: what if the first number entered was the highest possible value? This would then be the highest and lowest value to start)*

Initialize the low variable to 100, and the program then updates the low value as it should.

13. Set the low equal to that number.

14. Run the program again, and enter the values 10, 5, 15, and 8. Did it correctly report the high and low as 5 and 15? Yes.

15. Look at the main() method at the bottom of the code page. Add and remove comments so you are only running the Example3() code, then continue to the next section.

Example 3

This example shows the use of a loop to make a calculation. The user will enter a number. The program will then calculate the sum of the factors of a number. It will report that the number is Deficient, Abundant, or Perfect based upon a comparison of the sum of the factors to the number entered.

If the sum of the factors is greater than the number, then it is abundant. A value where the sum is less than the number is a deficient number. If the sum is equal to the number, it is considered Perfect. For example...

12 = 1 + 2 + 3 + 4 + 6 = 16, so 12 is an abundant number
16 = 1 + 2 + 4 + 8 = 15 = 16, so 16 is a deficient number
28 = 1 + 2 + 4 + 7 + 14 = 28, so 28 is a perfect number

1. The factors of 12 are 1, 2, 3, 4, and 6. The sum of the factors added together is 16. Run the program. Enter the value 12. Did you get the sum of 16? Yes.

2. Let's trace the program to see how this works. Look at the for loop. What value does the variable i start on? It starts at 1.

3. Given that num is 12, what will the variable i count up to? 11 because as long as i < num, it will increment.

4. Observe the if statement in the for loop. Describe the logic you see. In the for loop, it starts at 1 and goes to num (not including num, though). It then checks if modulus division results in a 0, if it does, this means it's a factor of the number so it uses sum += i to add it to the sum variable.

5. Now we need to add an additional if/if else structure. After the program displays the sum of the factors, add an if statement. Print out if the number is abundant, deficient, or perfect based upon the criteria described above.

6. Add an if statement to prevent the user from calculating a negative number.

7. Modify the program to add a loop. The loop should allow the user to keep entering numbers until they type -1 to quit. The value -1 will not be calculated as a number type, it will only quit the program. *Hint: Make sure to reset the sum variable to 0 for each iteration!*

8. Run the program to ensure it works.