Faculty of Science, Engineering and Technology

Introduction to Programming

Tutorial Task 3.1.2: Name Tester – Part 2

Overview

Control flow enables you to easily add conditions and loops to your programs. In this task you will create a small program that uses conditions and loops to output custom messages to users.

Purpose:	Learn to use the control flow statements within a program.
Task:	Create a program that tests a user's name and echoes a custom message.
Time:	This task should be completed before the start of week 4.
Resources:	▪ Chapters 6 and 8 of 'Learn to Program' **Chapt 6: Flow Control** **Chapt 8: Writing Your Own Methods**

Note: Remember to submit **all tasks** to Canvas. Also make sure you *fix and resubmit* any Pass tasks you did not get signed off last week!

Submission Details

You must submit the following files to Canvas:

- NameTester program source code.
- A screenshot of your program running
- Answers to questions about control flow.

Make sure that your task has the following in your submission:

- Demonstrates use of Ruby programming convention, including indentation within selection and repetition statements.
- Demonstrates use of an **if** statement to perform selection, and a **while** loop to perform repetition.

Instructions

Create a small program that will check the user's name and respond with different messages for different people.

Build on Part 1 to add a loop as follows:

Saying that the name is 'silly' will have a much greater effect if we add lots of 'silly's to the output... Add a **print_silly_name** procedure to your Name Test program. This will output the person's name and ' is a silly silly' ... with 60 'silly's, then ' name'.

Call this new procedure from main when the name is not your name.

The pseudocode for this procedure follows:

```
Procedure: print_silly_name
--------------------------------
Parameter:
 - name (the silly name String)
Local Variables:
 - i (an integer, used to count the number of loops)
--------------------------------
Make i equal 0
Print (staying on the same line) name, ' is a' , "\n"
While i is less than 60
do
  Print (on the same line) ' silly'
  Increment i (make i equal i + 1)
end
Output ' name!' (moving to a new line)
```

Note: Indentation is important! Notice that lines 4 and 5 in Listing 2 are indented... this means that they are inside the while loop. Line 6 is not indented so it sits outside the while loop, as the instruction after the loop.

Hint: Use **print** instead of **puts** to output without going to a new line.

The output of the program should look like this:

```
What is your name?
Sam
Sam is a
silly silly silly silly silly silly silly silly silly silly silly silly si
lly silly silly silly silly silly silly silly silly silly silly silly sill
y silly silly silly silly silly silly silly silly silly silly silly silly
silly silly silly silly silly silly silly silly silly silly silly silly si
lly silly silly silly silly silly silly name!
```

Get a screenshot of the terminal showing the out of running this program with your name, and with someone else's name.

End of Task