Faculty of Science, Engineering and Technology

Introduction to Programming

Tutorial Task 1.2: Desk Checking

Overview

This task will help you to learn about variables and to desk check and test a sequence of statements using variables.

Purpose: Demonstrate an understanding of sequence in programming code and practice checking code.

Task: Use the instructions on the following pages to desk check a program and test what it does. Write the code in Ruby, test it and submit it to Canvas when complete. Answer the questions provided.

Time: This task should be completed in your first lab.

Resources: **See Section 3.3 of the following book:**
Students' Guide to Program Design, by Lesley Anne Robertson (2014)
(click to access in Swinburne Library)

Submission Details

You must submit the following to Canvas:

- The desk checking for program 2 and answers to the questions provided (use the answer sheet in the Tasks Resources).
- A screenshot of running and testing the Ruby code provided.

SWINBURNE UNIVERSITY OF TECHNOLOGY

Instructions

1. Use desk checking to demonstrate that a program works as expected (Program 2 below).
2. See the example (Program 1) below and follow that format. Use the link in the 'Resources' section of this document (above) if you need further examples.
3. Complete the answer sheet (in the Task's Resources on Canvas) and when you finish upload as a document to Canvas.
4. Run the Ruby code provided for Program 2 (in the Task's Resources on Canvas) and check that the test results are what is expected.

Note: Remember these are commands that are executed in **sequence**. Each statement (line) **does something** then progam control moves to the next statement.

Program 1: Add 2 numbers *(Example of Desk Checking)*

Required Variables:

Integer: a, b, c.

Pseudocode:

Read the value of **a**

Read the value of **b**

Add **a** to **b** and assign the result to **c**

Print the value of **c** to the terminal.

Test Data:

	First data set	***Second data set***
a	***10***	***3***
b	***5***	***4***

Expected Result:

	First data set	***Second data set***
Output:	***15***	***7***

Desk check:

	Statement	**a**	**b**	**c**	**output**
First Pass	Read the value of **a**	10			
	Read the value of **b**		5		
	Add **a** to **b** and assign the result to **c**			15	
	Print the value of **c** to the terminal				15
Second Pass	Read the value of **a**	3			
	Read the value of **b**		4		
	Add **a** to **b** and assign the result to **c**			7	
	Print the value of **c** to the terminal				7

Program 2: Calculate Meal Total *(You do this one)*

Required Variables:

Integer: appetizer_price, main_price, dessert_price

Real (floating point): total_price

Pseudocode:

Read the value of appetizer_price

Read the value of main_price

Read the value of dessert_price

total_price = appetizer_price + main_price + dessert_price

Print '$' then the value of total_price to the terminal showing two decimal places.

Test Data:

	First data set	*Second data set*
appetizer_price	*10.30*	*12.40*
main_price	*34.00*	*41.00*
dessert_price	*8.50*	*9.80*

Expected Result:

	First data set	*Second data set*
Output:	*$52.80*	*$63.20*

Desk check - fill this in by completing the missing code in bill_total.rb (in the tasks Resources folder) then running it with the test data above:

	Statement	*appetizer_price*	*main_price*	*dessert_price*	*total_price*	*output*
First Pass	*Read the value of appetizer_price*	*10.30*				
	Read the value of main_price		*34.00*			
	Read the value of dessert_price			*8.50*		
	Calculate the total_price				*52.80*	
	Output the unit (dollars)					*$*
	Output the total_price					*52.80*
Second Pass	*Read the value of appetizer_price*					
	Read the value of main_price					
	Read the value of dessert_price					
	Calculate the total_price					
	Output the unit (dollars)					
	Output the total_price					

Complete the following desk checking and short answer questions on the answer sheet provided.

Question 1:

Desk check Program 2: Calculate Meal Total

	Statement	**appetizer _price**	**main _price**	**dessert _price**	**total _price**	**output**
First Pass	Read the value of **appetizer_price**					
	Read the value of **main_price**					
	Read the value of **dessert_price**					
	Calculate the **total_price**					
	Output the **total_price**					
Second Pass	Read the value of **appetizer_price**					
	Read the value of **main_price**					
	Read the value of **dessert_price**					
	Calculate the **total_price**					
	Output the **total_price**					

Q2: Short Answer Questions

Answer the following questions in the answer sheet provided in the resources for this task.

1. Using a few sentences explain why it may be important to execute statements in the correct sequence. (eg: what might happen if the last statement in Program 2 was executed earlier)

2: The code main_price = 10 is an example of which kind of programming statement?

3: What actions does the computer perform when it executes a = a + b?

4: How would the value of variable i change in the statement i = i + 1?

5: What sort of types will Ruby use to store the following variables (given the associated variable values)?

Data	Type
A person's name e.g: “Fred Smith”	
Number of students in a class e.g: 23	
Average age of a group of people e.g: 23.5	
A temperature in Celsius e.g: 45.7	
True or false e.g: 1 == 2	

6: Variables have a scope – what are two different scopes variables can have in Ruby?

Note: This is one of the tasks you need to **submit to Canvas**. Tutors should give guidance and perhaps feedback in the tutorial class.
Check the assessment criteria for the important aspect your tutor MAY check when assessing your finished portfolio.