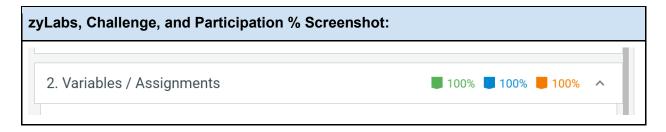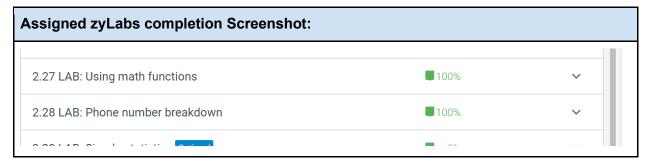# Assignment xx Algorithmic Design Document

Make a copy before you begin (File -> Make a copy). Add the Assignment # above and complete the sections below BEFORE you begin to code and submit with your Assignment to D2L (File -> Download -> PDF). The sections will expand as you type.

zyBooks

Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all assigned zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

zyLabs, Challenge, and Participation % Screenshot:
2. Variables / Assignments   100%   100%   100%   ^

Assigned zyLabs completion Screenshot:
2.27 LAB: Using math functions   100%   ⌄   2.28 LAB: Phone number breakdown   100%   ⌄

Assignment

Program description:
This program will take your Radon level and calculate the half-life radon levels at 4 days, 8 days, and 16 days from the initial measurement.

Before you begin coding, **you must first plan out the logic** and think about what data you will use to test your program for correctness. All programmers plan before coding - this saves a lot of time and frustration! Use the steps below to identify the inputs and outputs, calculations, and steps needed to solve the problem.

Algorithmic design:
a.   Identify all of the user input. What are the data types of the inputs? Define the input variables.
The radon detection level (integer or float, both are accepted)

The program will output the following in order:

Welcome message
Prompt requesting initial input
Radon input as <u>double</u> rounded to 4th decimal

Radon calc1 as <u>double</u> rounded to 4th decimal
Radon calc2 as <u>double</u> rounded to 4th decimal
Radon calc3 as <u>double</u> rounded to 4th decimal

End program message

radonInit = radonInit

radon half-life 1 = radonInit / 2.0
radon half-life 2 = radonInit / 4.0
radon half-life 3 = radonInit / 8.0

START

DECLARE double radonInit
DECLARE double radonH01
DECLARE double radonH02
DECLARE double radonH03

DISPLAY "Hello! Welcome to this Radon level calculator!"
DISPLAY "---"

DISPLAY "Please enter your starting Radon measurement in pCi/L: "
INPUT radonInit
DISPLAY "You have entered Radon level: %0.4lf pCi/L", radonInit

SET radonH01 = radonInit / 2.0
SET radonH02 = radonInit / 4.0
SET radonH03 = radonInit / 8.0

DISPLAY "Here are your Radon levels at future dates: "
DISPLAY "Radon levels at day 4: %0.4lf pCi/L", radonH01
DISPLAY "Radon levels at day 8: %0.4lf pCi/L ", radonH02

DISPLAY "Radon levels at day 16: %0.4lf pCi/L ", radonH03

DISPLAY "Thank you for using this Calculator."

END

---

e. Include 2 Sample Program Runs for your program using your own set of data. This data set must be different from my Sample Runs in the Assignment document. This process is similar to Unit Testing and will help you test your program better.

Sample Program Run 1:

Hello! Welcome to this Radon level calculator!"

Please enter your starting Radon measurement in pCi/L: **9**

You have entered Radon level: 9.0000 pCi/L

Here are your Radon levels at future dates:
Radon levels at day 4: 4.5000 pCi/L
Radon levels at day 8: 2.2500 pCi/L
Radon levels at day 16: 1.1250 pCi/L

Thank you for using this Calculator.

Sample Program Run 2:

Hello! Welcome to this Radon level calculator!"

Please enter your starting Radon measurement in pCi/L: **12.47569**

You have entered Radon level: 12.4757 pCi/L

Here are your Radon levels at future dates:
Radon levels at day 4: 6.2378 pCi/L
Radon levels at day 8: 3.1189 pCi/L
Radon levels at day 16: 1.5595 pCi/L

Thank you for using this Calculator.

Pseudocode Syntax

Think about each step in your algorithm as an action and use the verbs below:

To do this:	Use this verb:	Example:
Create a variable	DECLARE	`DECLARE integer num_dogs`

Print to the console window	DISPLAY	`DISPLAY "Hello!"`
Read input from the user into a variable	INPUT	`INPUT num_dogs`
Update the contents of a variable	SET	`SET num_dogs = num_dogs + 1`

Conditionals

Use a single alternative conditional	IF *condition* THEN
	    statement
	    statement
	END IF
		`IF num_dogs > 10 THEN`
		`    DISPLAY "That is a lot of dogs!"`
		`END IF`

Use a dual alternative conditional	IF *condition* THEN
	    statement
	    statement
	ELSE
	    statement
	    statement
	END IF
		`IF num_dogs > 10 THEN`
		`    DISPLAY "You have more than 10 dogs!"`
		`ELSE`
		`    DISPLAY "You have ten or fewer dogs!"`
		`END IF`

Use a switch/case statement	SELECT *variable or expression*
	  CASE *value_1:*
	    statement
	    statement
	  CASE *value_2:*
	    statement
	    statement
	  CASE *value_2:*
	    statement
	    statement
	  DEFAULT:
	    statement
	    statement
	END SELECT
		`SELECT num_dogs`
		`   CASE 0: DISPLAY "No dogs!"`
		`   CASE 1: DISPLAY "One dog.."`
		`   CASE 2: DISPLAY "Two dogs.."`
		`   CASE 3: DISPLAY "Three dogs.."`
		`   DEFAULT: DISPLAY "Lots of dogs!"`
		`END SELECT`

Loops

Loop while a condition is true - the loop body will execute 0 or more times.	WHILE *condition*
	    statement
	    statement
	END WHILE
		`SET num_dogs = 1`
		`WHILE num_dogs < 10`
		`    DISPLAY num_dogs, " dogs!"`
		`    SET num_dogs = num_dogs + 1`
		`END WHILE`

Loop while a condition is true - the loop body will execute 1 or more times.	DO
	    statement
	    statement
	WHILE *condition*
		`SET num_dogs = 1`
		`DO`
		`    DISPLAY num_dogs, " dogs!"`
		`    SET num_dogs = num_dogs + 1`
		`WHILE num_dogs < 10`

Loop a specific number of times.	FOR *counter* = *start* TO *end*
	    statement
	    statement
	END FOR
		`FOR count = 1 TO 10`
		`    DISPLAY num_dogs, " dogs!"`
		`END FOR`

Functions		
Create a function	FUNCTION *return_type* *name (parameters)* *statement* *statement* END FUNCTION	```FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION```
Call a function	CALL *function_name*	```CALL add(2, 3)```
Return data from a function	RETURN *value*	```RETURN 2 + 3```