CS 161A: Programming and Problem Solving I

Assignment 2 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. The sections will expand as you type. When you are finished, download this document as a PDF (File -> Download -> PDF) and submit to D2L.

This document contains an interactive checklist. To mark an item as complete, click on the box (the entire list will be highlighted), then right click (the clicked box will only be highlighted), and choose the checkmark.

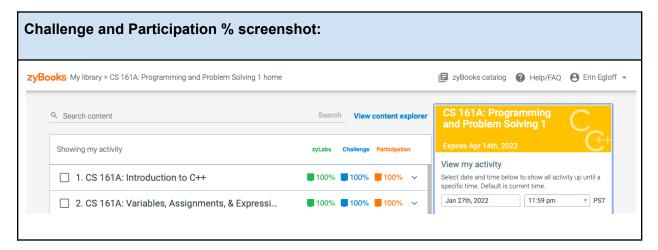
Planning your program before you start coding is part of the development process. In this document you will:

Paste a screenshot of	your zyBool	ks Challenge an	d Participation %

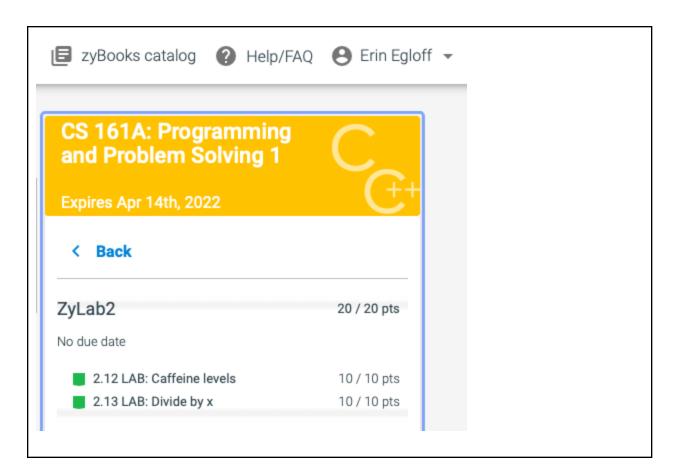
- ☐ Paste a screenshot of your assigned zyLabs completion
- Write a detailed description of your program, at least two complete sentences
- ☐ If applicable, design a sample run with test input and output
- ☐ Identify the program inputs and their data types
- ☐ Identify the program outputs and their data types
- ☐ Identify any calculations or formulas needed
- ☐ Write the algorithmic steps as pseudocode or a flowchart
- ☐ Tools for flowchart Draw.io Diagrams.net

1. 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%.



Assigned zyLabs completion screenshot:



2. Program Description

In the box below, describe the purpose of the program. You must include a detailed description with at least two complete sentences.

Program description:

The purpose of this program is to calculate the number of gumballs in a jar. Using the volume of the jar, the jar's capacity, and the radius of a gumball, we should be able to estimate the total number of gumballs.

3. Sample Run

If you are designing your own program, you will start with a sample run. Imagine a user is running your program - what will they see? What inputs do you expect, and what will be the outputs from the given inputs? Choose test data you will use to test your program. Calculate and show the expected outputs. Use the sample run to test your program.

Sample run:

"Welcome to my program that estimates the number of gumballs in the jar!"

"Enter the radius of one gumball (cm) and the volume of the jar (mL) separated by a space:"

(user inputs values)

Ex. 1 100

"There are approximately 15 gumballs in the jar."

4. Algorithmic Design

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 and list all of the user input and their data types.

Radius of gumball, Volume of jar

b. Identify and list all of the user output and their data types.

Number of gumballs

c. What calculations do you need to do to transform inputs into outputs? List all formulas needed, if applicable. If there are no calculations needed, state there are no calculations for this algorithm.

Volume of a gumball = $4.0 / 3 \pi r^3$

Total number of gumballs = .64 (% of jar volume) / volume of a gumball

Design the logic of your program using pseudocode or flowcharts. Here is where you
would use conditionals, loops or functions (if applicable) and list the steps in transforming

inputs into outputs. Walk through your logic steps with the test data from the assignment document or the sample run above.

- 1. DECLARE integer radius_gumball
- 2. DECLARE integer volume gumball
- 3. DECLARE integer volume_jar
- 4. DECLARE integer LOAD_FACTOR = 0.64
- 5. DECLARE integer total gumballs
- 6. DISPLAY "What is the radius of one gumball (in cm) and the volume of the jar (in mL) separated by a space?"
- 7. INPUT radius gumball
- 8. INPUT volume_jar
- 9. SET volume gumball = $(4.0 / 3) \pi (radius gumball)^3$
- 10. SET volume_jar *= LOAD_FACTOR
- 11. SET total_gumballs = volume_jar / volume_gumball
- 12. DISPLAY "There are approximately " total_gumballs " in the jar."

5. 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	onditionals			
Use a single alternative conditional	IF condition THEN statement statement END IF	<pre>IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF</pre>		
Use a dual alternative conditional IF condition THEN statement statement ELSE statement		IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or		

	statement END IF	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 CASE value_2: statement CASE value_1: statement 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	<pre>SET num_dogs = 1 WHILE num_dogs < 10 DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 END WHILE</pre>			
Loop while a condition is true - the loop body will execute 1 or more times.	DO statement statement WHILE condition	<pre>SET num_dogs = 1 DO DISPLAY num_dogs, " dogs!" SET num_dogs = num_dogs + 1 WHILE num_dogs < 10</pre>			
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			