CS 161A: Programming and Problem Solving I

Assignment A05 Sample 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:

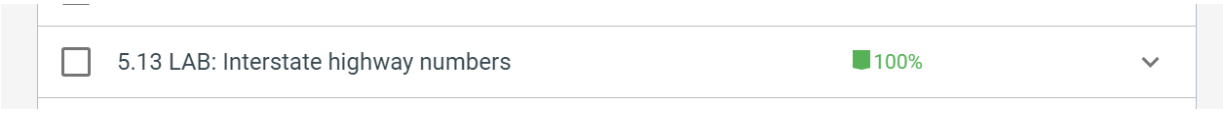
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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%.

Challenge and Participation % screenshot:

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:
This program will let you enter two lines of test that are checked against each other and substituted to make fun outputs! It will also detect if these strings lines are matching, or don't contain matching words/phrases between them

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 the Phrases and Subphrases program!

Enter Phrase 1: I am almost done here and can go outside soon :)
You entered: I am almost done here and can go outside soon :)
Enter Phrase 2: and can
You entered: and can

and can is found in I am almost done here and can go outside soon :)
and can go outside soon :)

Thank you for using my program!
```

```
Welcome to the Phrases and Subphrases program!

Enter Phrase 1: thisisatest
You entered: thisisatest
Enter Phrase 2: thisisatest
You entered: thisisatest

Both phrases match

Thank you for using my program!
```

```
Welcome to the Phrases and Subphrases program!

Enter Phrase 1: I really like trains
```

```
You entered: I really like trains
Enter Phrase 2: Airplanes are cool too
You entered: Airplanes are cool too

No match

Thank you for using my program!
```

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.
INPUT phrase1 as Str INPUT phrase2 as Str
b. Identify and list all of the user output and their data types.
String: userInput for phrase1 String: userInput for phrase2 String Phrase match; phrase not found in phrase; substring of phrase
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.
Substring start = IndexFoundStart + lenPhraseShort Phrase1 > phrase2; phrase1 < phrase2; phrase1 == phrase2
d. 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.

```
START


DECLARE String phrase1 = ""

DECLARE String phrase2 = ""

DECLARE String phraseLong = ""

DECLARE String phraseShort = ""

DECLARE String mergedPhrase = ""

DECLARE lenPhraseShort = 0

DECLARE indexFoundStart = 0

DECLARE Int runState = 0



DISPLAY "Welcome to the Phrases and Subphrases program!"


DISPLAY "Enter Phrase 1: "

INPUT String phrase1

DISPLAY "You entered: {phrase1}"



DISPLAY "Enter Phrase 2: "

INPUT String phrase2

DISPLAY "You entered: {phrase2}"
```

```
IF phrase1 = phrase 2
- - DISPLAY "Both phrases match"
- - SET runState = 1
ELSE IF lenPhrase1 > lenPhrase2
- - SET phraseLong = phrase1
- - SET phraseShort = phrase2
- - SET lenPhraseShort = phrase2
ELSE
- - SET phraseLong = phrase2
- - SET phraseShort = phrase1
- - SET lenPhraseShort = phrase1

IF phraseShort NOT IN phraseLong w/ STRING::NPOS
- - DISPLAY "No match"
- - runState = 1



IF runState != 1
- - SET indexFoundStart = phraseLong.find(phraseShort)
- - mergedPhrase = phraseShort + substring(phraseLong + lenPhraseShort)
- - DISPLAY "{phraseShort} is found in {phraseLong}"
- - DISPLAY "{mergedPhrase}"

DISPLAY Thank you for using my program!
```

END

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		

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
		`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`

	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`