CS 161A/B: Programming and Problem Solving I

Algorithm 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.

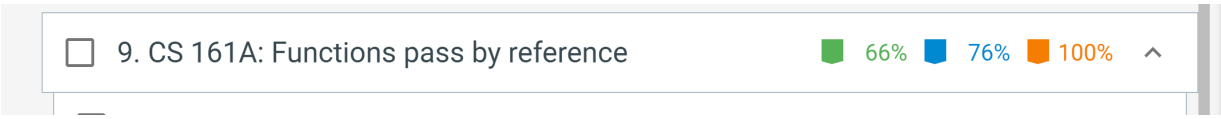
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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

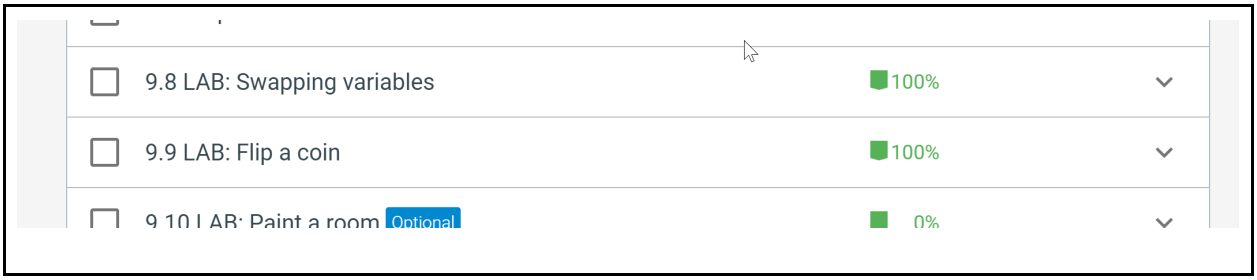
1. zyBooks

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 allow you to enter all your assignment scores, your midterm and final scores, and get a final calculation of your class grade on a 4.0 scale.

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 Final Grade Calculator! Please enter the following information and I will calculate your Final Numerical Grade and Letter Grade for you! The number of assignments must be between 0 and 10. All scores entered must be between 0 and 4.   Enter the number of assignments (0 to 10): **3** Enter score 1: **2** Enter score 2: **1**

Enter score 3: **3**

Enter your midterm exam score: **-5**

Enter your final exam score: **3**

Illegal Value! Please try again!!

Enter your midterm exam score: **3**

Enter your final exam score: **2**

Your Final Numeric score is **2.2**

Your Final Grade is C

Thank you for using my Grade Calculator!

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.

Use the pseudocode syntax shown in the document, supplemented with English phrases if necessary. **Do not include any implementation details (e.g. source code file names, class or struct definitions, or language syntax)**. Do not include any C++ specific syntax or data types.

Algorithmic design:
a. Identify and list all of the user input and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string (for CS161B and up).
Input: Int numAssignments

Int userInt # Double userDouble
b. Identify and list all of the user output and their data types. Include a variable name, data type, and description. Data types include string, integer, floating point, (single) character, and boolean. Data structures should be referenced by name, e.g. “array of integer” or “array of string” (for CS161B and up).
Output: Double gradeAverage # Double finalGradeLetter
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. Formulae should reference the variable names from step a and step b as applicable.

assignAvgBase100 = (assignAvg / 4.0) * ASSIGNMENT_WEIGHT;
midtermBase100 = (midterm / 4.0) * EXAM_WEIGHT;
finalBase100 = (final / 4.0) * EXAM_WEIGHT;
finalGrade = (assignAvgBase100 + midtermBase100 + finalBase100) / 100;
finalGrade = finalGrade * 4;

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. **Use the syntax shown at the bottom of this document and plain English phrases. Do not include any implementation details (e.g. file names) or C++ specific syntax.**
START DECLARE CONST Int ASSIGNMENT_WEIGHT = 60 DECLARE CONST Int EXAM_WEIGHT = 20

```
DECLARE Int Function main()
- - DECLARE Int numAssignments
- - DECALRE Double assignmentAvg
- - DECLARE midtermScorre
- - DECLARE finalScore
- - DECLARE gradeAverage
- - DECLARE String scorePrompt
- - DECLARE Char finalGRadeLeter
- - DECLARE Int i
- -
- - SET PRECISSION = 2
- -
- - CALL welcome()
- -
- - DO
- - - - SET numAssignments = CALL readInt("Enter num of assignments 0 - 10")
- - - - IF numAssignments < 0 OR numAssignments > 10
- - - - - - CALL errorCode(0)
- - WHILE numAssignment < 0 OR numAssignments > 10
- -
- - SET assignmentAvg = CALL assignAverage(numAssignments)
- -
- - DO
- - - - CALL readScore(midtermScore, finalScore)
```

```
- - - -
- - - - IF midtermScore < 0 OR midtermScore > 4
- - - - - - CALL errorCode(0)
- - - - IF finalScore < 0 OR finalScore > 4
- - - - - - CALL errorCode(0)
- - WHILE finalScore < 0 OR finalScore > 4 OR midtermScore < 0 OR midtermScore > 4
- -
- - SET gradeAverage = CALL calcFinalScore(assignmentAvg, midtermScore, finalScore)
- -
- - SET finalGradeLetter = CALL calcLetterGrade(gradeAverage)
- -
- - DISPLAY "Final numerica score is {gradeAverage}"
- - DISPLAY "Final grade is {finalGRadeLetter}"
- -
- - DISPLAY "Thank you and goodbye"
- -
- - RETURN 0



DECLARE Void Function welcome()
- - DISPLAY "Welcome to my Final Grade Calculator!"
- - DISPLAY "Please enter the following information and I will calculate your "
- - DISPLAY "Final Numerical Grade and Letter Grade for you!"
- - DISPLAY "The number of assignments must be between 0 and 10."
- - DISPLAY "All scores entered must be between 0 and 4."
```

```
DECLARE Void Function errorCode(Int errorNum)
- - IF errorNum == 0
- - - - DISPLAY "Invalid data was given"
- - ELSE
- - - - DISPLAY "Super error, alert the dummy who wrote that code"



DECLARE Void Function consoleClear()
- - CLEAR CONSOLE
- - INPUT IGNORE



DECLARE Int Function readInt(string prompt)
- - DECLARE Int userInt
- -
- - DISPLAY "{prompt}"
- - INPUT userInt
- -
- - CALL consoleClear()
- - RETURN userInt



DECLARE Double Function readScore(string prompt, double& refVar)
```

```
- - DECLARE Int userDouble
- -
- - DISPLAY "{prompt}"
- - INPUT userDouble
- -
- - CALL consoleClear()
- - SET refVar = userDouble



void getInput(&Double midtermScore, Double& finalExamScore)
- - CALL readScore("Enter your midterm exam score: ", midtermExamScore)
- - CALL readScore("Enter your midterm exam score: ", finalExameScore)


DECLARE Double Function assignAverage(int numAssigns)
- - DECLARE String scorePrompt
- - DECLARE Double newScore
- - DECLARE Double totalScore
- - DECLARE Double calcAverage
- - DECLARE Int i
- -
- - FOR i = 0 WHILE i < numAssigns, ++i
- - - - DO
- - - - - - SET scorePrompt = "Enter score {i + 1}"
- - - - - - SET newScore = CALL readScore(scorePrompt)
- - - - - -
```

```
- - - - - - IF newScore < 0 OR newScore > 4
- - - - - - - - CALL errorCode(0)
- - - - - - ELSE
- - - - - - - - totalScore = totalScore + newScore
- - - - WHILE newScore < 0 OR newScore > 4
- -
- - SET calcAverage = totalScore / numAssigns
- - RETURN calcAverage



DECLARE Double Function calcFinalScore(Double assignAvg, Double midterm, Double final)
- - DECLARE Double assignAvgBase100
- - DECLARE Double midtermBase100
- - DECALRE finalBase100
- - DECLARE finalGrade
- -
- - SET assignAvgBase100 = (assignAvg / 4.0) * ASSIGNMENT_WEIGHT
- - SET midtermBase100 = (midterm / 4.0) * EXAM_WEIGHT
- - SET finalBase100 = (final / 4.0) * EXAM_WEIGHT
- -
- - SET finalGrade = (assignAvgBase100 + midtermBase100 + finalBase100) / 100
- - SET finalGrade = finalGrade * 4
- -
- - RETURN finalGrade
```

```
DECALRE Void Function calcLetterGrade(Double finalScore, Char& letter)
- - DECLARE Char letterGrade
- -
- - IF finalScore >= 3.3
- - - - SET letterGrade = 'A'
- - ELSE IF finalScore < 3.3 AND finalScore >= 2.8
- - - - SET letterGRade = 'B'
- - ELSE IF finalScore < 2.8 AND finalScore >= 2.0
- - - - SET letterGRade = 'C'
- - ELSE IF finalScore < 2.0 AND finalScore >= 1.2
- - - - SET letterGRade = 'D'
- - ELSE
- - - - SET letterGRade = 'F'
- -
- - SET letter = letterGRade


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
	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
		`FOR count = 1 TO 10`
		`  DISPLAY num_dogs, " dogs!"`
		`END FOR`

	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`