CS 161A: Programming and Problem Solving I

Assignment A08 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:

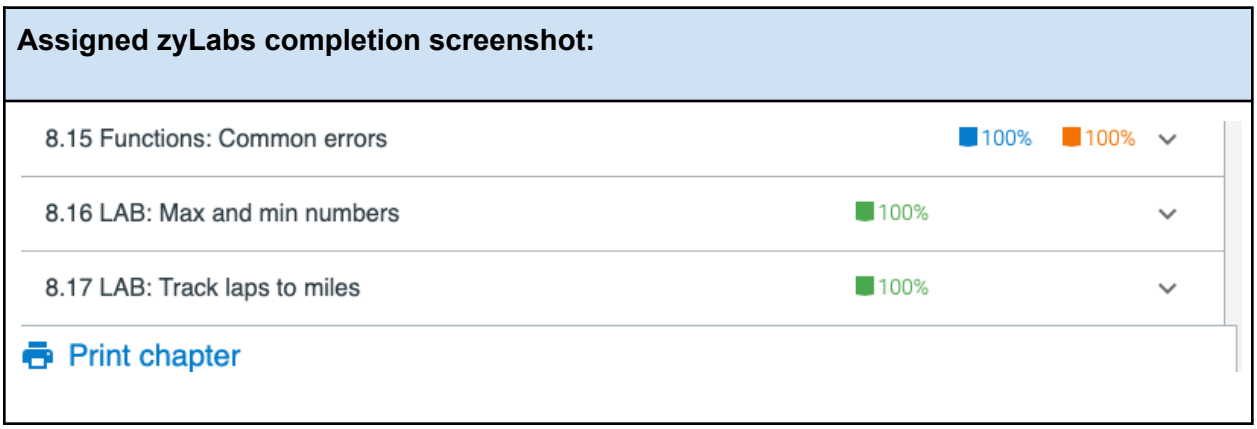
- ❏ Paste a screenshot of your zyBooks Challenge and Participation %
- ❏ Paste a screenshot of your assigned zyLabs completion
- ❏ Write a detailed description of your program, at least two complete sentences
- ❏ If applicable, design a sample run with test input and output
- ❏ Identify the program inputs 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 is designed to calculate my final score of the letter grade for anyone in my CS161A class with Dona Hertel. This calculation will include scores from assignments, exams, and discussions using weights of the grade.

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): 6

Enter score for assignment 1: 3.4
Enter score for assignment 2: 4
Enter score for assignment 3: 2.5
Enter score for assignment 4: 3.3
Enter score for assignment 5: 3.1
Enter score for assignment 6: 2.5

Enter your midterm exam score: 3.5

Enter your final exam score: 4

Your Final Numeric score is 3.4
Your Final Grade is A

Thank you for using my Grade Calculator!
```

```
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 for assignment 1: 3
Enter score for assignment 2: 4
Enter score for assignment 3: 2.5

Enter your midterm exam score: 2.5

Enter your final exam score: 2

Your Final Numeric score is 2.8
```

```
Your Final Grade is B

Thank you for using my Grade Calculator!
```

```
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): 12
Illegal Value! Please try again!!
Enter the number of assignments (0 to 10): 5

Enter score for assignment 1: 3.4
Enter score for assignment 2: 4
Enter score for assignment 3: 2.5
Enter score for assignment 4: 5.5
Illegal Score! Please try again!
Enter score for assignment 4: 3.5
Enter score for assignment 5: 3.1

Enter your midterm exam score: 3.5

Enter your final exam score: 4

Your Final Numeric score is 3.5
Your Final Grade is A

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.

Algorithmic design:
a. Identify and list all of the user input and their data types.
○ input as **integer** (to input the non-negative integer for # of assignments) ○ score as **double** (for our grade in decimal points form)
b. Identify and list all of the user output and their data types.

- Functions:
    1. **const double ASSIGNMENT_WEIGHT = 0.60**
    2. **const double EXAM_WEIGHT = 0.20**
    3. **void WelcomeMessage**()
    4. **integer ReadInt** (const string & prompt)
    5. **double ReadScore** (const string & prompt)
    6. **double AssignAverage** (**integer** numAssigns)
    7. **double CalcFinalScore** (**double** assignAvg, **double** midterm, **double** finalExam)
    8. **char CalcLetterGrade** (**double** finalScore)
- MAIN()
    1. **integer** numAssigns = **ReadInt** function(string)
    2. **double** assignAvg = **AssignAverage** function (numAssigns)
    3. **double** midterm = **ReadScore** function (string)
    4. **double** finalExam = **ReadScore** function (string)
    5. **char** letterGrade = **CalcLetterGrade** **(**finalScore)
    6. **double** totalScore = 0.0

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.

- **double CalcFinalScore**(**double** assignAvg, **double** midterm, **double** finalExam)
    1. Return (assignAvg * **ASSIGNMENT_WEIGHT**) + (midterm + **EXAM_WEIGHT**) + (finalExam * **EXAM_WEIGHT**)

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.

1. **DECLARE const double ASSIGNMENT_WEIGHT** = 0.60
2. **DECLARE const double EXAM_WEIGHT** = 0.20
3. **FUNCTION void WelcomeMessage**()
4. **FUNCTION integer ReadInt** (const string & prompt)
5. **FUNCTION double ReadScore** (const string & prompt)
6. **FUNCTION double AssignAverage** (**integer** numAssigns)
7. **FUNCTION double CalcFinalScore** (**double** assignAvg, **double** midterm, **double** finalExam)
8. **FUNCTION char CalcLetterGrade** (**double** finalScore)e)
9. **FUNCTION MAIN**
    a. **CALL WelcomeMessage**

b. **DECLARE integer** numAssigns = **ReadInt** function(string)
c. **DECLARE double** assignAvg = **AssignAverage** function (numAssigns)
d. **DECLARE double** midterm = **ReadScore** function (string)
e. **DECLARE double** finalExam = **ReadScore** function (string)
f. **DECLARE char** letterGrade = **CalcLetterGrade** (finalScore)
g. **DISPLAY fixed setprecision(1)**
h. DISPLAY string + finalScore
i. DISPLAY string + letterGrade

10. END FUNCTION Main()

11. **CALL void WelcomeMessage**()
    a. **DISPLAY** welcome message

12. **CALL ReadInt()**
    a. **DECLARE** input
    b. DO
        i. **DISPLAY** prompt
        ii. **INPUT** input
        iii. **IF** input < 0 or input > 10 or NOT cin
            1. **DISPLAY** invalid input
            2. **DISPLAY** clear cin
            3. **DISPLAY** ignore cin 10000
        iv. END IF
    c. **WHILE** input < 0 or input > 10 or NOT cin
        i. **RETURN** input

13. **END FUNCTION ReadInt()**

14. **CALL ReadScore()**
    a. **DECLARE** score
    b. DO
        i. **DISPLAY** prompt
        ii. **INPUT** score
        iii. **IF** score < 0, OR score > 4 OR NOT cin
            1. **DISPLAY** invalid message
            2. **DISPLAY** clear cin
            3. **DISPLAY** ignore cin 10000
        iv. **WHILE** score < 0, score > 4, NOT cin
        v. **Return** score
    c. END DO WHILE

15. **END FUNCTION ReadScore()**

16. **CALL AssignAverage**(int numAssigns)
    a. **DECLARE** totalScore = 0.0
    b. **FOR** int i = 0, i < numAssigns, i++
        i. totalScore = ReadScore()(string + to_string(i+1) + ": "
    c. **END** FOR LOOP
    d. **RETURN** totalScore / numAssigns

17. **END FUNCTION** **AssignAverage**
18. **CALL** **CalcFinalScore**(double assignAvg, double midterm, double finalExam)
    a. **RETURN**
        i. **CALCULATE** assignAvg * **ASSIGNMENT_WEIGHT** + midterm * **EXAM_WEIGHT** + finalExam * **EXAM_WEIGHT**
19. **END FUNCTION** **CalcFinalScore**
20. **CALL** **FUNCTION** **CalcLetterGrade**(double finalScore)
    a. **IF** finalScore >= 3.3
        i. return A
    b. **ELSE IF** finalScore >= 2.8
        i. return B
    c. **ELSE IF** finalScore >= 2.0
        i. return C
    d. **ELSE IF** finalScore >= 1.2
        i. return D
    e. **ELSE**
        i. return F
21. **END FUNCTION** **CalcLetterGrade**
22. **END PROGRAM**

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*	`IF num_dogs > 10 THEN` `  DISPLAY "You have more than`

	statement
	ELSE
	statement
	statement
	END IF
		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