

# 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](https://draw.io) - [Diagrams.net](https://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:**

9. CS 161A: Functions pass by reference		 100%  100%  100% 
9.1 Pass by Reference - Examples and Hand Trace	No activities	
9.2 Pass by reference	 100%  100% 	
9.3 Scope of variable/function definitions	 100% 	
9.4 Default parameter values	 100%  100% 	
9.5 Function name overloading	 100%  100% 	
9.6 Parameter error checking	 100% 	
9.7 Preprocessor and include	 100% 	

Assigned zyLabs completion screenshot:		
8.15 Functions: Common errors	 100%  100% 	
8.16 LAB: Max and min numbers	 100% 	
8.17 LAB: Track laps to miles	 100% 	
 <a href="#">Print chapter</a>		

## 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.

<b>Program description:</b>
<p>This program is designed to calculate a final</p> <p style="padding-left: 40px;">score of the letter grade for anyone in CS161A.</p> <p style="padding-left: 40px;">This calculation will include scores from</p>

assignments and exams using weighted 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 1: 3.4
Enter score 2: 4
Enter score 3: 2.5
Enter score 4: 3.3
Enter score 5: 3.1
Enter score 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 1: 3
Enter score 2: 4
Enter score 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 1: 3.4

Enter score 2: 4

Enter score 3: 2.5

Enter score 4: 5.5

Illegal Score! Please try again!

Enter score 4: 3.5

Enter score 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.

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).

b. **input** as **integer** (to input the non-negative integer for # of assignments)

c. **num** as **double** (for our grade in decimal points form)

d. 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).

e. Functions:

a. **const double ASSIGNMENT\_WEIGHT = 0.60**

b. **const double EXAM\_WEIGHT = 0.20**

c. **void WelcomeMessage()**

d. **integer ReadInt** (const string& prompt)

e. **void ReadScore** (const string& prompt, double &num)

f. **double AssignAverage** (**integer numAssigns**)

g. **void** GetInput(double &midtermScore, double &finalExamScore

h. **double CalcFinalScore** (**double assignAvg**, **double midterm**, **double finalExamScore**)

i. **void CalcLetterGrade** (**double finalScore**, **char &letter**)

f. MAIN()

a. **integer numAssigns** = **ReadInt** function(string)

b. **double assignAvg** = **AssignAverage** function (**numAssigns**)

c. **double midtermScore**

d. **double finalExamScore**

e. **char letterGrade**

f. **double totalScore** = 0.0

g. 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.

*Calculating total score before weighted*

totalScore = totalScore + score

*Calculating total number of assignments larger than 0*

if numAssigns > 0

- totalScore divided by numAssigns
- ELSE 0.0

*Calculating the final score using the weights (below):*

```
double CalcFinalScore(double assignAvg, double midtermScore,
double finalExamScore)
{
    return (assignAvg * ASSIGNMENT_WEIGHT) + (midtermScore *
        EXAM_WEIGHT) + (finalExamScore * EXAM_WEIGHT);
}
```

- h. 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.**

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 void ReadScore (const string & prompt, double &num)
6. FUNCTION double AssignAverage (integer numAssigns)
7. FUNCTION void GetInput (double &midtermScore, double &finalExamScore)
8. FUNCTION double CalcFinalScore (double assignAvg, double midterm, double finalExamScore)
9. FUNCTION void CalcLetterGrade (double finalScore, char &letter)
10. FUNCTION MAIN
  - a. CALL WelcomeMessage
  - b. DECLARE integer numAssigns = ReadInt function(string)
  - c. DECLARE double assignAvg = AssignAverage function (numAssigns)
  - d. DISPLAY 2 end lines
  - e. DECLARE double midtermScore = ReadScore function (string)
  - f. DECLARE double finalExamScore = ReadScore function (string)
  - g. CALL GetInput (midtermScore, finalExamScore)
  - h. DISPLAY 2 end lines
  - i. DECLARE double finalScore = CalcFinalScore (assignAvg, midtermScore, finalExamScore)
  - j. DECLARE char letterGrade
  - k. CALL CalcLetterGrade(finalScore, letterGrade)
  - l. DISPLAY fixed setprecision(1)
  - m. DISPLAY string + finalScore
  - n. DISPLAY string + letterGrade
  - o. DISPLAY string thank you message
11. END FUNCTION Main()
12. CALL void WelcomeMessage()

```

    a. DISPLAY welcome message
13. 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
14. END FUNCTION ReadInt()
15. CALL ReadScore(const string& prompt, double &num)
    a. DO
        i. DISPLAY prompt
        ii. INPUT num
        iii. IF num < 0, OR num > 4 OR NOT cin
            1. DISPLAY invalid message
            2. DISPLAY clear cin
            3. DISPLAY ignore cin 10000
        iv. WHILE num < 0, num > 4, NOT cin
    b. END DO WHILE
16. END FUNCTION ReadScore()
17. CALL AssignAverage(int numAssigns)
    a. DECLARE totalScore = 0.0
    b. FOR int i = 0, i < numAssigns, i++
        i. DECLARE score (double)
        ii. ReadScore()(string + to_string(i+1) + ": ", score)
        iii. SET totalScore = totalScore + score
    c. END FOR LOOP
    d. RETURN IF numAssigns > 0,
        a. totalScore/numAssigns
        b. ELSE 0.0
18. END FUNCTION AssignAverage
19. CALL GetInput(double &midtermScore, double &finalExamScore)
    a. ReadScore(string, midtermScore)
    b. ReadScore(string, finalExamScore)
20. END FUNCTION
21. CALL CalcFinalScore(double assignAvg, double midtermScore, double
    finalExamScore)
    a. RETURN

```

```

        i.   CALCULATE assignAvg * ASSIGNMENT_WEIGHT + midtermScore *
              EXAM_WEIGHT + finalExamScore * EXAM_WEIGHT
22. END FUNCTION CalcFinalScore
23. CALL FUNCTION CalcLetterGrade(double finalScore, char &letter)
    a. IF finalScore >= 3.3
        i.   letter = 'A'
    b. ELSE IF finalScore >= 2.8
        i.   letter = 'B'
    c. ELSE IF finalScore >= 2.0
        i.   letter = 'C'
    d. ELSE IF finalScore >= 1.2
        i.   letter = 'D'
    e. ELSE
        i.   letter = 'F'
24. END FUNCTION CalcLetterGrade
25. 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
<b>Conditionals</b>		
Use a single alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> END IF	IF num_dogs > 10 THEN DISPLAY "That is a lot of dogs!" END IF
Use a dual alternative conditional	IF <i>condition</i> THEN <i>statement</i> <i>statement</i> ELSE <i>statement</i> <i>statement</i>	IF num_dogs > 10 THEN DISPLAY "You have more than 10 dogs!" ELSE DISPLAY "You have ten or fewer dogs!"



	END IF	END IF
Use a switch/case statement	SELECT <i>variable or expression</i> CASE <i>value_1</i> : <i>statement</i> CASE <i>value_2</i> : <i>statement</i> CASE <i>value_2</i> : <i>statement</i> CASE <i>value_2</i> : <i>statement</i> DEFAULT: <i>statement</i> <i>statement</i> 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
<b>Loops</b>		
Loop while a condition is true - the loop body will execute 0 or more times.	WHILE <i>condition</i> <i>statement</i> <i>statement</i> 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 <i>statement</i> <i>statement</i> WHILE <i>condition</i>	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 <i>counter</i> = <i>start</i> TO <i>end</i> <i>statement</i> <i>statement</i> END FOR	FOR count = 1 TO 10 DISPLAY num_dogs, " dogs!" END FOR
<b>Functions</b>		
Create a function	FUNCTION <i>return_type</i> <i>name (parameters)</i> <i>statement</i> <i>statement</i> END FUNCTION	FUNCTION Integer add(Integer num1, Integer num2) DECLARE Integer sum SET sum = num1 + num2 RETURN sum END FUNCTION
Call a function	CALL <i>function_name</i>	CALL add(2, 3)
Return data from a function	RETURN <i>value</i>	RETURN 2 + 3