CS 161B: Programming and Problem Solving I

Assignment A01 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.

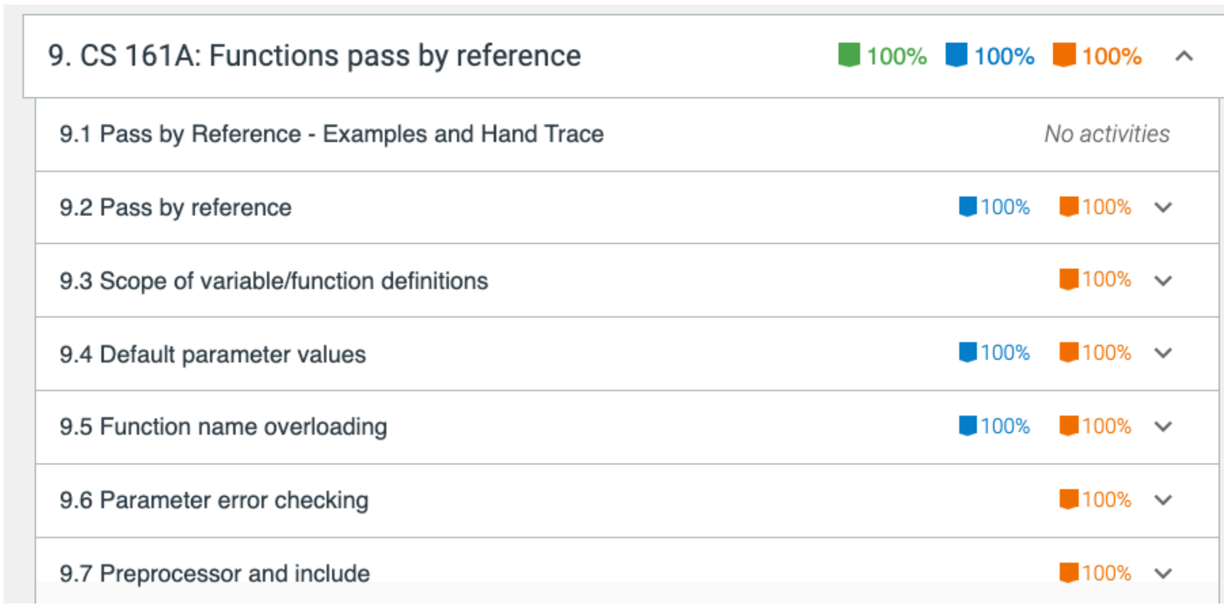
Planning your program before you start coding is part of the development process. In this document you will:

- ■ 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%.

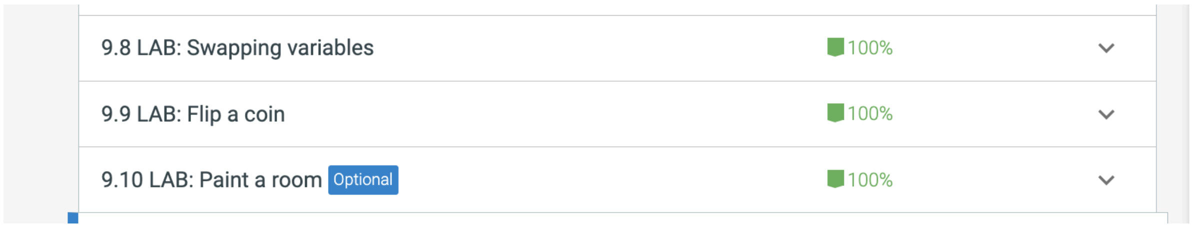
Challenge and Participation % screenshot:



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:



9.8 LAB: Swapping variables	100%
9.9 LAB: Flip a coin	100%
9.10 LAB: Paint a room Optional	100%

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 C++ program will calculate the price for a customer's order, using menu with two choices: input a menu item name and price, or QUIT the program. The user will be able to select many choices until they quit the program manually. The code will be using several functions, a do-while loop, and an exit condition without breaks or returns in the loop.

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 Food Cart Program!
What would you like to do today?
Pick an option from below:
        1. Place an order
        2. Quit
>> 9
Invalid Option! Please choose 1-2!
>> 1
Enter the name of your item: Pasta
Enter the cost of your item $: 15.75
Do you want another item? (y/n): y
Enter the name of your item: Bowl
Enter the cost of your item $: 12.75
Do you want another item? (y/n): y
Enter the name of your item: Soda
Enter the cost of your item $: 3.50
Do you want another item? (y/n): x
Invalid Option! Please choose y/n!
>> n

Your total is: $32.00

Enter the amount of tip you want to add $: 3.50

Your total is: $35.50
You get a 5% discount!
Your discount is $1.78
Your final total is: $33.73

What would you like to do today?
Pick an option from below:
        1. Place an order
        2. Quit

>> 1
Enter the name of your item: Fajita Bowl
Enter the cost of your item $: 20.75
Do you want another item? (y/n): y
Enter the name of your item: Vietnamese plate
Enter the cost of your item $: 22.75
```

```
Do you want another item? (y/n): y
Enter the name of your item: Soda
Enter the cost of your item $: 3.50
Do you want another item? (y/n): x
Invalid Option! Please choose y/n!
>> n

Your total is: $47.00

Enter the amount of tip you want to add $: 10.00

Your total is: $57.00
You get a 10% discount!
Your discount is $5.70
Your final total is: $51.30


What would you like to do today?
Pick an option from below:
        1. Place an order
        2. Quit

>> 2

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.
○ **num** as integer (to read the input from user menu in readInt function and again in readDouble function) ○ **itemName** as string (to read menu item name via input) ○ **anotherItem** as string (to read another menu item name via input)
b. Identify and list all of the user output and their data types.
○ **option** as integer (to hold a menu item choice number)

- **totalCost** as double (to hold total plus tip plus discount)
- **tip** as double (to hold the tip amount)
- **discount** as double (to hold discount amount, and calculate it)
- **finalTotal** as double (to hold complete cost of tip, discount and total)
- **total** as double (to hold the total)
- **anotherItem** as double (to hold the tip amount)
- **prompt** as char (to hold a menu choice of y/n)

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.

```
○ SET totalCost = totalCost + tip
○ SET finalTotal = totalCost - discount
○ SET discount = total * DISCOUNT_AMOUNT1
○ SET discount = total * DISCOUNT_AMOUNT2
○ SET total = cost + tip
```

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.  CREATE FUNCTION [void] welcome()
    END FUNCTION welcome()
2.  CREATE FUNCTION [void] displayMenu()
    END FUNCTION displayMenu()
3.  CREATE FUNCTION [void] readOption(int &option)
    END FUNCTION readOption()
4.  CREATE FUNCTION [void] readInt(string prompt, int &num)
    END FUNCTION readInt()
5.  CREATE FUNCTION [void] readDouble(string prompt, double &num)
    END FUNCTION readDouble()
6.  CREATE FUNCTION [void] placeOrder(double &cost)
    END FUNCTION placeOrder()
7.  CREATE FUNCTION double tipDiscount(double &tip, double &discount, double cost)
8.  END FUNCTION tipDiscount()
9.  DECLARE const double DISCOUNT1 = 50.0
10. DECLARE const double DISCOUNT2 = 35.0
```

```
11. DECLARE const double DISCOUNT_AMOUNT1 = 0.10
12. DECLARE const double DISCOUNT_AMOUNT2 = 0.05
13. CREATE FUNCTION  main()
    a. DECLARE integer option
    b. DECLARE double totalCost = 0.0
    c. DO WHILE
        i.   IF totalCost is 0
                1. CALL welcome()
        ii.  END IF
        iii. CALL displayMenu()
        iv.  CALL readOption(option)

        v.   IF (option = 1) THEN
                1. CALL placeOrder(totalCost)
                2. DECLARE double tip
                3. DECLARE double discount

                4. DISPLAY Your total is message

                5. CALL readDouble (enter tip string, tip)
                6. SET totalCost = totalCost + tip
                7. SET totalCost = CALL tipDiscount(tip, discount, totalCost)

                8. DISPLAY Your total is message

                9. IF (discount > 0) THEN
                    a. DISPLAY "You get a discount" message
                        i. use ternary statement here for discount
                    b. DISPLAY "Your discount : $ " + discount
                    c. END IF
                10. DECLARE finalTotal = totalCost - discount
                11. DISLPLAY final total message
                12. SET totalCost to 0 (to reset for the next loop iteration)

        vi.  ELSE IF (option = 2)
                    a. DISPLAY "Thank you for using my program!"
```

```
            vii.  ELSE
                  1. DISPLAY “Invalid Option! Please choose 1-2!”
            viii. END IF
      d. WHILE option does not equal 2
      e. END DO WHILE
      f. return 0;
14. END FUNCTION main()
15. RETURN FUNCTION welcome()
      a. DISPLAY “Welcome to my Food Cart Program!”
      b. DISPLAY “What would you like to do today?”
16. END FUNCTION welcome()
17. RETURN FUNCTION displayMenu()
      a. DISPLAY “Pick an option from below:”
      b. DISPLAY “1. Place an order”
      c. DISPLAY “2. Quit”
      d. DISPLAY “>> “
18. END FUNCTION displayMenu()
19. RETURN FUNCTION readOption(int &option)
      a. CALL readInt(“”, option)
      b. WHILE (option < 1 or option > 2)
            i.   DISPLAY “Invalid OPtion! Please choose 1-2!”
            ii.  readInt(“”, option)
20. END FUNCTION readOption()
21. RETURN FUNCTION readInt(string prompt, int &num)
      a. while (true)
            i.   DISPLAY prompt
            ii.  IF no input for num
                  1. CLEAR BUFFER
                  2. IGNORE the input
                  3. DISPLAY “Invalid input. Please enter a valid number.”
            iii. ELSE
                  1. IGNORE the input
                  2. break
            iv.  END IF
      b. END WHILE
22. END FUNCTION readInt()
```

```
23. RETURN FUNCTION readDouble(string prompt, double &num)
    a. WHILE (true)
        i. DISPLAY prompt
        ii. IF there is no input for num
            1. CLEAR BUFFER
            2. IGNORE the input
            3. DISPLAY “Invalid input. Please enter a valid number.”
        iii. ELSE
            1. IGNORE the input
            2. break
24. END FUNCTION readDouble()
25. RETURN FUNCTION placeOrder(double &cost)
    a. DECLARE char anotherItem
    b. DO
        i. DECLARE string itemName
        ii. DECLARE double itemCost

        iii. DISPLAY “Enter the name of your item: “
        iv. GET ENTIRE INPUT [getline(cin, itemName)]
        v. CALL readDouble(“Enter the cost of your item $: “, itemCost)

        vi. cost = cost + itemCost

        vii. DISPLAY “Do you want another item? (y/n): “
        viii. INPUT anotherItem
        ix. IGNORE the input

        x. WHILE (tolower(anotherItem) is NOT ‘y’ && tolower(anotherItem) is
           NOT ‘n’
            1. DISPLAY “Invalid Option! Please choose y/n!”
            2. DISPLAY “Do you want another item? (y/n): “
            3. INPUT anotherItem
            4. IGNORE the input
        xi. END WHILE
    c. WHILE (tolower(anotherItem is ‘y’)
    d. END DO WHILE
```

```
26. END FUNCTION placeOrder()
27. RETURN FUNCTION tipDiscount(doublt &tip, double &discount, double cost)
    a. CALL readDouble("Enter the amount of tip you want to add $: ", tip)
    b. DECLARE double total = cost + tip
    c. IF total is greater than DISCOUNT_AMOUNT1
        i. discount = total * DISCOUNT_AMOUNT1
    d. ELSE IF total is greater than DISCOUNT_AMOUNT2
        i. discount = total * DISCOUNT_AMOUNT2
    e. ELSE
        i. discount = 0.0
    f. END IF
    g. RETURN total - discount
28. END FUNCTION tipDiscount()
```

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
	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`