CS 161B: Programming and Problem Solving I

Assignment A01 Algorithmic Design Document

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.

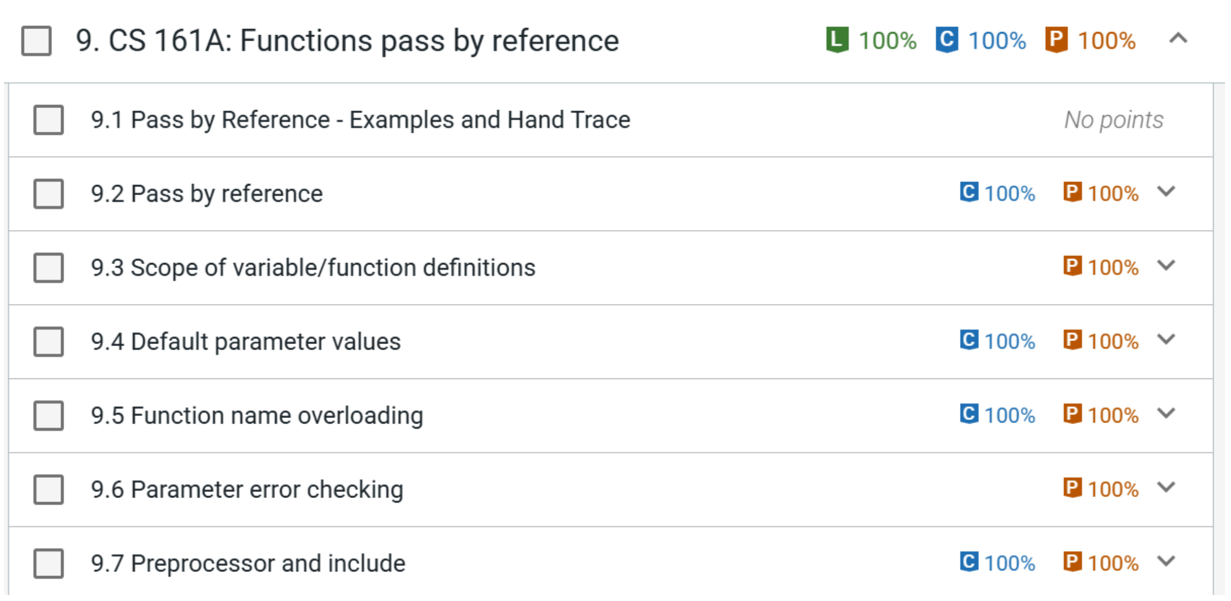
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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

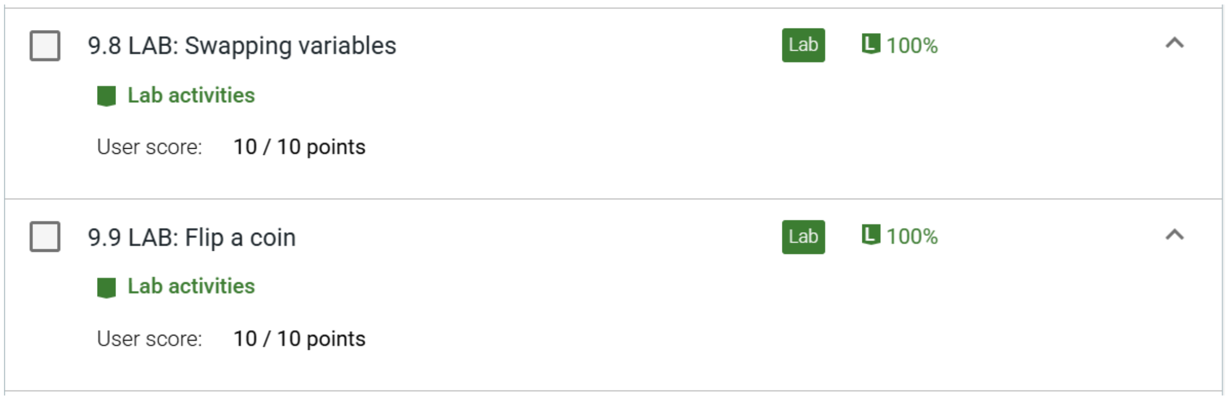
Add your zyBooks screenshots for the % and assigned zyLabs completions below. Required percentages: all **assigned** zyLabs, Challenge Activity with at least 70%, and Participation Activity with at least 80%.

Challenge and Participation % screenshot:



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

Assigned zyLabs completion screenshot:



9.8 LAB: Swapping variables Lab 100%
Lab activities
User score: 10 / 10 points

9.9 LAB: Flip a coin Lab 100%
Lab activities
User score: 10 / 10 points

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 act as a cashier by welcoming a user. The proceeding to ask if they would like to order, take the order and add up the cost

Ends by prompting the user to enter a tip and adding that to total. Checking if a discount is applicable before returning the total cost with applicable discount.

Looping until user decides to end the program

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:

```
Hello there and welcome to my food cart!
What would you like to order?
```

```
Pick one of the options below:
1. Place an Order
2. Quit
Option: 9
Invalid input! Please enter 1 or 2:
1
Enter the name of the item: Pasta
Enter the cost of your item: $15.75
Do you want another item (y/n)? y
Enter the name of the item: Bowl
Enter the cost of your item: $12.75
Do you want another item (y/n)? y
Enter the name of the item: Soda
Enter the cost of your item: $3.50
Do you want another item (y/n)? x
Invalid input! Please enter 'y' for yes or 'n' for no :
n

Your total is: $32.00
Enter a tip amount: $3.50

Your total is: $35.50
You get a 5.00% discount!
You discount today is $1.78

Your final total is: $33.73

Pick one of the options below:
1. Place an Order
2. Quit
Option: 1
Enter the name of the item: Fajita Bowl
Enter the cost of your item: $20.75
Do you want another item (y/n)? y
Enter the name of the item: Vietnamese plate
Enter the cost of your item: $22.75
Do you want another item (y/n)? y
Enter the name of the item: soda
Enter the cost of your item: $3.50
Do you want another item (y/n)? x
Invalid input! Please enter 'y' for yes or 'n' for no :
n

Your total is: $47.00
Enter a tip amount: $10.00

Your total is: $57.00
You get a 10.00% discount!
You discount today is $5.70
```

```
Your final total is: $51.30

Pick one of the options below:
1. Place an Order
2. Quit
Option: 2
Goodbye!
```

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.
○ Int userOption ○ Double cost ○ Double tip ○ Char userOrder ○ Double localCost ○ Char userOrder ○ String menuItem
b. Identify and list all of the user output and their data types.
○ Double cost (Cost of order) ○ Double discount (Order discount if applicable)
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 cost += localcost ○ SET total += tip ○ SET total *= (1 - discount) ○ DISPLAY discount * 100 ○ DISPLAY cost * discount

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. FUNCTION welcome()
      a. Hello there and welcome to my food cart
      b. What would you like to order
   END FUNCTION welcome()

2. FUNCTION displayMenu()
      a. Pick options below
      b. 1. Order
      c. 2. Quit

   END FUNCTION displayMenu()

3. FUNCTION readOption(option)
      a. DISPLAY option:
      b. CALL readInt
      c. WHILE option is not 1 or 2
            i.   CALL readInt

   END FUNCTION readOption()

4. FUNCTION readInt(string prompt, int& num)
      a. DECLARE int tempVar to 0
      b. DISPLAY prompt
      c. INPUT tempVar
      d. WHILE NOT cin
            i.   DISPLAY error
            ii.  CLEAR input
            iii. INPUT tempVar
      e. SET num to tempVar

   END FUNCTION readInt()

5. FUNCTION readChar(string prompt, int& option)
      a. DISPLAY prompt
      b. INPUT option
      c. WHILE NOT cin
            i.   DISPLAY error
            ii.  CLEAR input
            iii. INPUT option
```

```
    END FUNCTION readChar()

6.  FUNCTION readDouble(string prompt, int& num)
        a. DECLARE double temp to 0
        b. DISPLAY prompt
        c. INPUT temp
        d. WHILE NOT cin
               i.   DISPLAY error
               ii.  CLEAR input
               iii. INPUT temp
        e. SET num to temp

    END FUNCTION readDouble()

7.  FUNCTION placeOrder(double& cost)
        a. DECLARE char userOrder to x
        b. DECLARE double localCost to 0
        c. DECLARE string prompt1 to “Enter the cost of your item”
        d. DECLARE string prompt2 to “Do you want another item?”
        e. While userOrder is NOT n or N
               i.   DISPLAY “Enter name of item”
               ii.  DECLARE string menuItem
               iii. INPUT menuItem
               iv.  CALL readDouble
               v.   SET cost to localCost
               vi.  CALL readChar

    END FUNCTION placeOrder()

8.  FUNCTION discountDisplay(double discount, double cost)
        a. DISPLAY “You get a” (discount)
        b. DISPLAY “Your discount today is $” (cost * discount)

    END FUNCTION discountDisplay()

9.  FUNCTION displayTotal(double cost)
        a. DISPLAY “Your total is: $” cost

    END FUNCTION displayTotal()

10. FUNCTION main()
        a. DECLARE int userOption
        b. DECLARE double cost
        c. DECLARE double discount
        d. DECLARE double tip
        e. CALL welcome()
        f.  DO LOOP
                i.    CALL displayMenu()
               ii.    CALL readOption
              iii.    IF userOption not 1
                          1.  CALL placeOrder
                          2.  CALL displayTotal
                          3.  SET cost AND CALL tipDiscount
                          4.  DISPLAY total
        g.  END DO WHILE userOption not 2
    11. DISPLAY "goodbye"
    12. END FUNCTION main()
```

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
		`SELECT num_dogs`
		`   CASE 0: DISPLAY "No dogs!"`
		`   CASE 1: DISPLAY "One dog.."`
		`   CASE 2: DISPLAY "Two dogs.."`

	statement
	CASE *value_2:*
	statement
	statement
	CASE *value_2:*
	statement
	statement
	DEFAULT:
	statement
	statement
	END SELECT
		`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`