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.

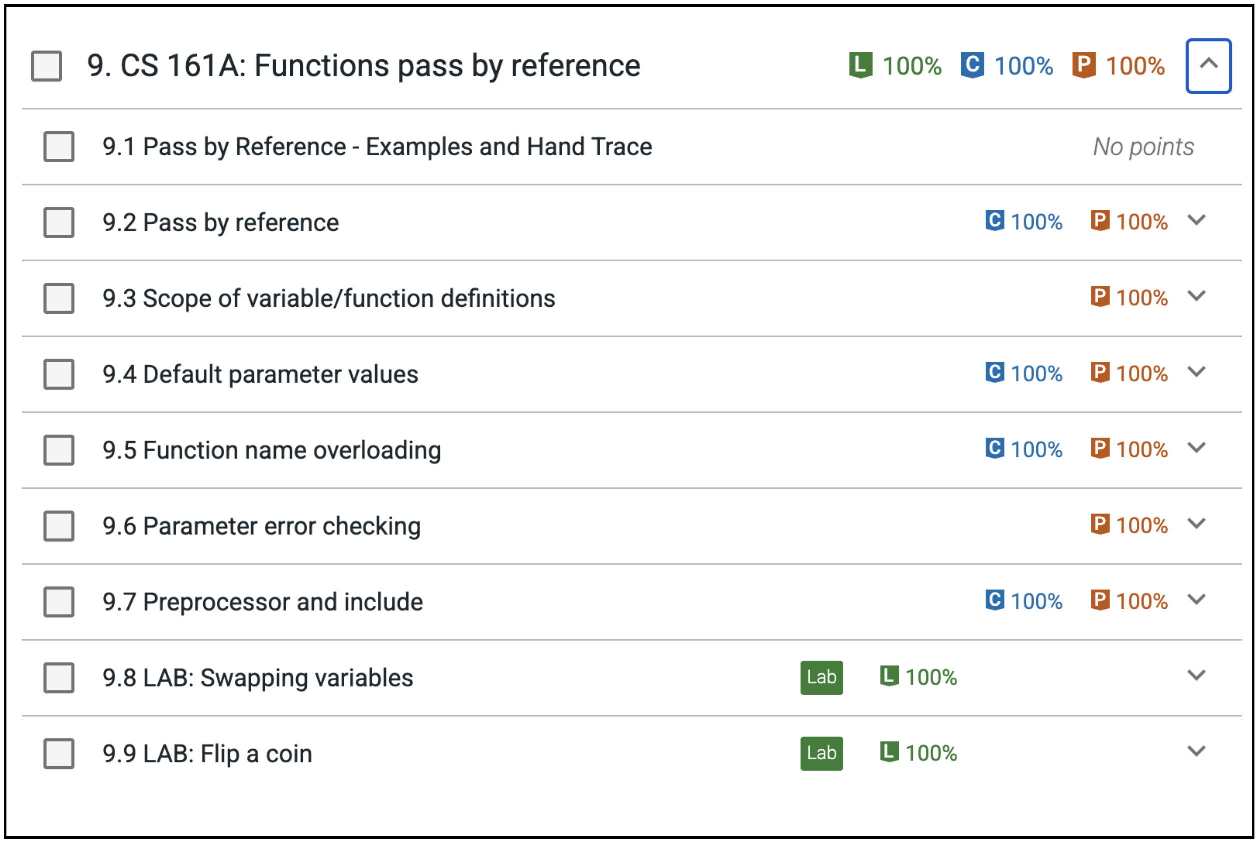
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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:



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%
9.8 LAB: Swapping variables	Lab L 100%
9.9 LAB: Flip a coin	Lab L 100%

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 prompt the user for an input if they would like to place an order or quit. If they choose to place an order, the the program will ask them for the name of what they ate and how much it cost. Then the program will calculate their cost to dine, tip, and add a discount if they reach a total cost threshold.

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!
Pick an option from below:
1. Place an order
2. Quit
>> 1

Enter the name of your item: Sushi
Enter the cost of your item: $45.45

Would you like to add another item? (y/n): n

Your total is: $45.45

Enter the amount of tip you want to add: $15.25

Your total is: $60.70
You get a 10% discount!
Your discount is: $6.07
Your final total is: $54.63

Pick an option from below:
1. Place an order
2. Quit
>> 1

Enter the name of your item: Sapporo
Enter the cost of your item: $8.00

Would you like to add another item? (y/n): n

Your total is: $8.00

Enter the amount of tip you want to add: $2.75

Your total is: $10.75
Your final total is: $10.75
```

```
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.

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).
Option(½) - integer - menu choice selected by the user Item - string - name of the food item Itemcost - floating point - cost of each individual item Cost - floating point - total of all items entered Tip - floating point - tip amount entered by user Option(y/n) - character - user choice to continue adding more items
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).

Cost - floating point - total of item costs before tip and discount

Tip - floating point - tip amount added to the order

Discount - floating point - discount amount calculated based on total

finalTotal floating point - Final total after tip and discount

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.

Total before discount:
Total = cost + tip

Discount calculation:
If total > 50:
Discount = total * 0.10
Else if total > 35:
Discount = total * 0.05
Else:
Discount = 0

Final total:
finalTotal = total - 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.
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.

```
FUNCTION Welcome()
DISPLAY "Welcome to my Food Cart Program!"
END FUNCTION


FUNCTION displayMenu()
DISPLAY "Pick an option:"
DISPLAY "1. Place an order"
DISPLAY "2. Quit"
END FUNCTION


FUNCTION readInt(prompt, num as int by reference)
LOOP until valid input
DISPLAY prompt
READ num
IF input is invalid
CLEAR input and retry
END FUNCTION


FUNCTION readDouble(prompt, num as double by reference)
LOOP until valid input
DISPLAY prompt
READ num
IF input is invalid or < 0
CLEAR input and retry
END FUNCTION


FUNCTION readOption (option as int by reference)
CALL readInt with prompt “>> “
WHILE option is not 1 or 2
DISPLAY error and retry
END FUNCTION
```

```
FUNCTION placeOrder(cost as double by reference)
SET itemCost = 0.0
LOOP
ASK for item name
ASK for item cost (use readDouble)
ADD item cost to total cost
ASK if user wants to add another item (y/n)
VALIDATE response until user says ‘n’
END FUNCTION



FUNCTION tipDiscount(tip as double by reference, discount as double by reference, cost as
double) returns double
ASK user for tip amount(use readDouble)
SET total = cost + tip
IF total > 50, discount = total * 0.10
ELSE IF total > 35 and < 50, discount = total * 0.05
RETURN total - discount
END FUNCTION




FUNCTION MAIN
MAIN()
FORMAT output to two decimal places
CALL welcome()
DO
CALL displayMenu()
CALL readOption(userChoice)
IF userChoice is 1
SET cost, tip, and discount to 0
CALL placeOrder(cost)
DISPLAY cost
CALL tipDiscount() and capture final total
DISPLAY total + tip, discount, and final total
WHILE userChoice != 2
CALL goodbye()
END 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
	    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`