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.

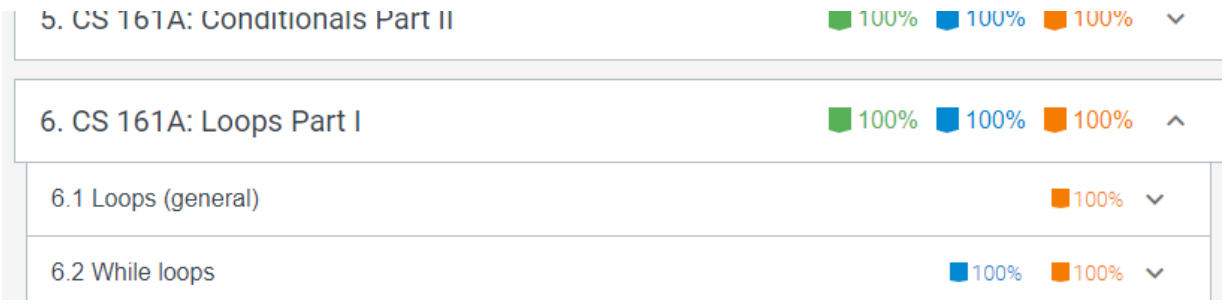
Planning your program before you start coding is part of the development process. In this document you will:

- ❏ 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

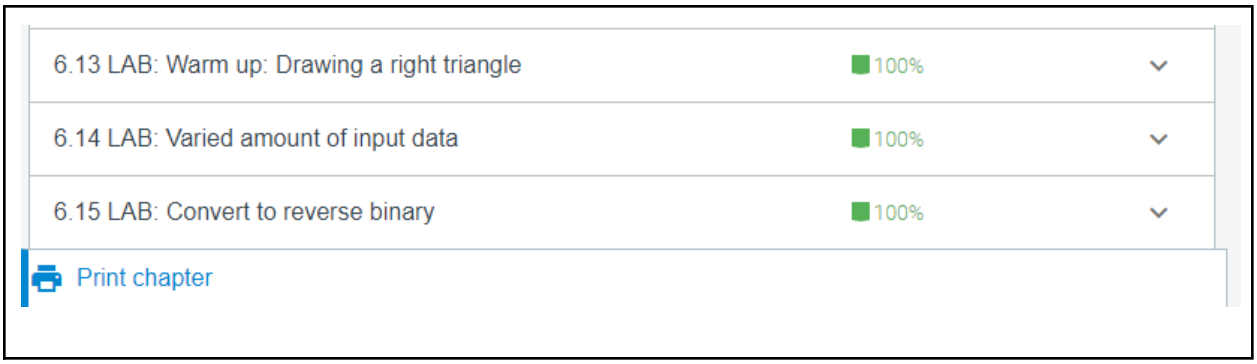
1. zyBooks

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:



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 help organize and calculate pizza for a pizza party. The program will ask for amounts of the number of people attending, average number of slices per person, and the cost of one pizza. The program's constant variables will calculate for them how much pizza and money it will take for the pizza party, as well as how many slices each person needs using accumulator variables.

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 Gina's Pizza Party Calculator! ***** Enter the number of people, average number of slices per person, and the cost of a pizza separated by a space: **10 2.6 10.50** Number of pizzas: 4 Cost of pizzas: $42.00 Tax: $2.94 Delivery: $8.99 Total Cost: $53.93

Do you want to enter more (y/n): **y**

Enter the number of people, average number of slices per person, and the cost of a pizza separated by a space: **9 2.5 10.95**

Number of pizzas: 3
Cost of pizzas: $32.85
Tax: $2.30
Delivery: $7.03
Total Cost: $42.18

Do you want to enter more (y/n): **y**

Enter the number of people, average number of slices per person, and the cost of a pizza separated by a space: **14 3.2 14.95**

Number of pizzas: 6
Cost of pizzas: $89.70
Tax: $6.28
Delivery: $19.20
Total Cost: $115.17

Do you want to enter more (y/n): **n**

Number of entries: 3
Total number of pizzas: 13
Average number of pizzas: 4.3
Maximum number of people: 14
Maximum cost of pizzas: $115.17

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).

numPeople (integer)

avgSlicesPerPerson (double)

costPerPizza (double)

userChoice (string)

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).

totalCost (double), **numEntries** (int), **totalPizzas**(int), **maxPeople** (int), **maxCost** (int),

SLICES_PER_PIZZA (const int)

SALES_TAX_RATE (const double)

DELIVERY_CHARGE (const double)

pizzasNeeded (integer)

pizzaCost (double)

tax (double)

deliveryCharge (double)

total (double)

avgPizzas (double)

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.

remainingSlices = **numPeople** * **avgSlicesPerPerson**;

```
pizzaCost = pizzasNeeded * costPerPizza

tax = pizzaCost * SALES_TAX_RATE

deliveryCharge = (pizzaCost + tax) * DELIVERY_CHARGE

total = pizzaCost + tax + deliveryCharge

avgPizzas = static_cast<double>(totalPizzas) / numEntries
```

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.

```
DECLARE & SET SLICES_PER_PIZZA = 8 (integer constant)
DECLARE & SET SALES_TAX_RATE = 0.07 (double constant)
DECLARE & SET DELIVERY_CHARGE = 0.20 (double constant)
DECLARE & SET numEntries = 0; (integer)
DECLARE & SET totalPizzas = 0; (integer)
DECLARE & SET maxPeople = 0; (integer)
DECLARE & SET totalCost = 0.0; (double)
DECLARE & SET maxCost = 0.0; (double)
DECLARE userChoice (string literal)
DISPLAY welcome message
DO
   DECLARE numPeople (integer)
   DECLARE avgSlicesPerPerson (double)
   DECLARE costPerPizza (double)
   DISPLAY # for user prompt
   INPUT numPeople >> avgSlicesPerPerson >> costPerPizza
   DECLARE & SET pizzasNeeded = 0 (integer)
   DECLARE & SET remainingSlices = numPeople * avgSlicesPerPerson;
   WHILE (remainingSlices >= SLICES_PER_PIZZA)
      THEN pizzasNeeded++;
      THEN remainingSlices -= SLICES_PER_PIZZA
   END WHILE
   IF (remainingSlices > 0)
      THEN pizzaNeeded++
   END IF
   DECLARE & SET pizzaCost (double) = pizzasNeeded * costPerPizza
   DECLARE & SET tax (double) = pizzaCost * SALES_TAX_RATE
   DECLARE & SET deliveryCharge (double)= (pizzaCost + tax) * DELIVERY_CHARGE
   DECLARE & SET total (double) = pizzaCost + tax + deliveryCharge
   SET totalPizzas += pizzasNeeded
   SET totalCost += total
```

```
   IF numPeople > maxPeople
      THEN SET maxPeople = numPeople
   END IF
   IF total > maxCost
      THEN SET maxCost = total
   END IF
   DISPLAY fixed << setprecision(2) → setting up to show 2 decimal places
   DISPLAY Number of Pizzas message + pizzasNeeded
   DISPLAY Cost of Pizzas $ + pizzaCost
   DISPLAY Tax $ + tax
   DISPLAY Delivery $ + deliveryCharge
   DISPLAY Total Cost $ + total
   DO
      DISPLAY asking user if they want to continue prompt
      INPUT userChoice
      IF userChoice is NOT y/n/Y/N/yes/no/YES/NO
         DISPLAY Invalid choice message & prompt again
      END IF
   WHILE userChoice is NOT y/n/Y/N/yes/no/YES/NO
      numEntries++
   END WHILE
WHILE userChoice is NOT y/Y/yes/YES
[END OF ORIGINAL DO WHILE^^^]

DECLARE & SET avgPizzas = static_cast<double>(totalPizzas) / numEntries
DISPLAY number of entries message + numEntries
DISPLAY total number of pizzas message + totalPizzas
DISPLAY average number of pizzas message + avgPizza (fixed<<setprecision(1))
DISPLAY max number of people message + maxPeople
DISPLAY maximum cost of pizza message + maxCost

DISPLAY exit message
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
	  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`