Assignment xx 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 and submit with your Assignment to D2L (File -> Download -> PDF). The sections will expand as you type.

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%.

zyLabs, Challenge, and Participation % Screenshot:

About this material

1. Introduction to C   🟩 100% 🟦 100% 🟧 100% ⌄

Assigned zyLabs completion Screenshot:

1.16 zyLab training: Basics	🟩100%	⌄
1.17 zyLab training: Interleaved input / output	🟩100%	⌄
1.18 LAB: Formatted output: Hello World!	🟩100%	⌄
1.19 LAB: Formatted output: No parking sign	🟩100%	⌄
1.20 LAB: Input and formatted output: Right-facing arrow	🟩100%	⌄

Assignment

Program description:
This program will take the number of large and small pizza's you are ordering, calculate how much the total order will be, and provide an itemized cost breakdown.

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:

We will require 2 inputs;
1. A whole number of large pizza(s) intended to be purchased
2. A whole number of small pizza(s) intended to be purchased

The program will output an arithmetic sum calculated from the supplied input values

Simple multiplication:

Small pizza cost = numSmall * costSmall
Large pizza cost = numLarge * costLarge

Simple arithmetic:
Total pizzas = numSmall + numLarge

START

DECLARE Int costSmall = 5
DECLARE Int costLarge = 9
DECLARE Int numSmall = 0
DECLARE Int numLarge = 0
DECLARE Int sumTotal = 0

DISPLAY "Hello! Welcome to the pizza cost calculator."

DISPLAY "Please input the number of small pizza's being ordered: "
INPUT Int numSmall
DISPLAY "Please input the number of large pizza's being ordered: "
INPUT Int numLarge

SET Int sumTotal = (numSmall * costLarge) + (numSmall * costLarge)

DISPLAY "You are ordering ", numSmall, " small pizzas"
DISPLAY "You are ordering ", numLarge, " large pizzas"

DISPLAY "Order total: $", sumTotal
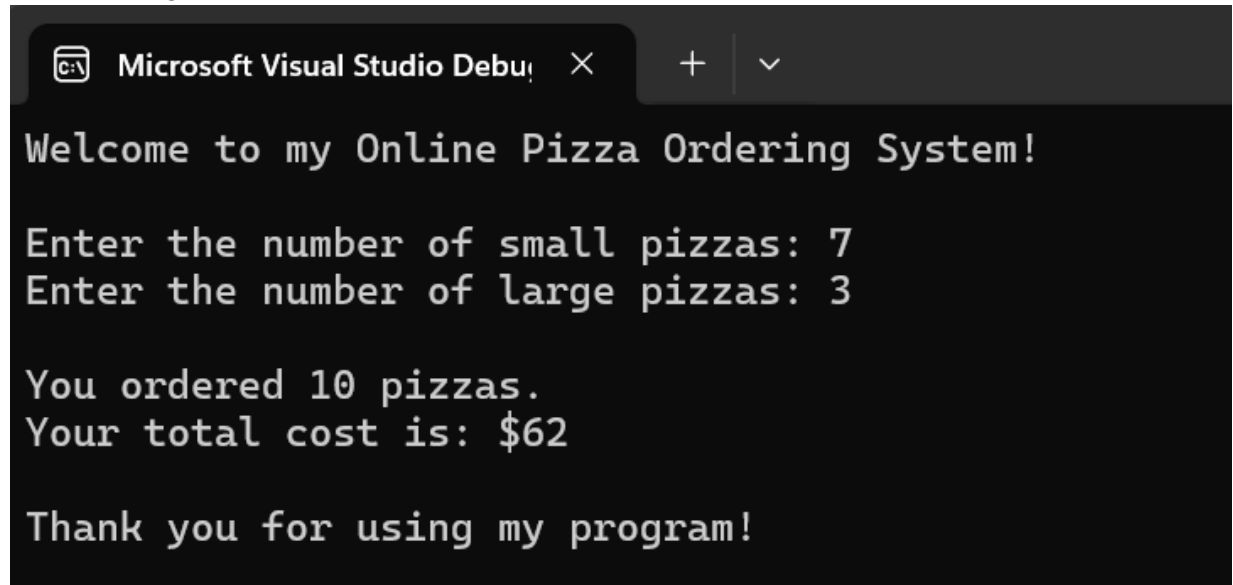DISPLAY numSmall, " pizza(s) at $5/unit is: ", (numSmall * costSmall)

DISPLAY numLarge, " pizza(s) at $9/unit is: ", (numLarge * costLarge)

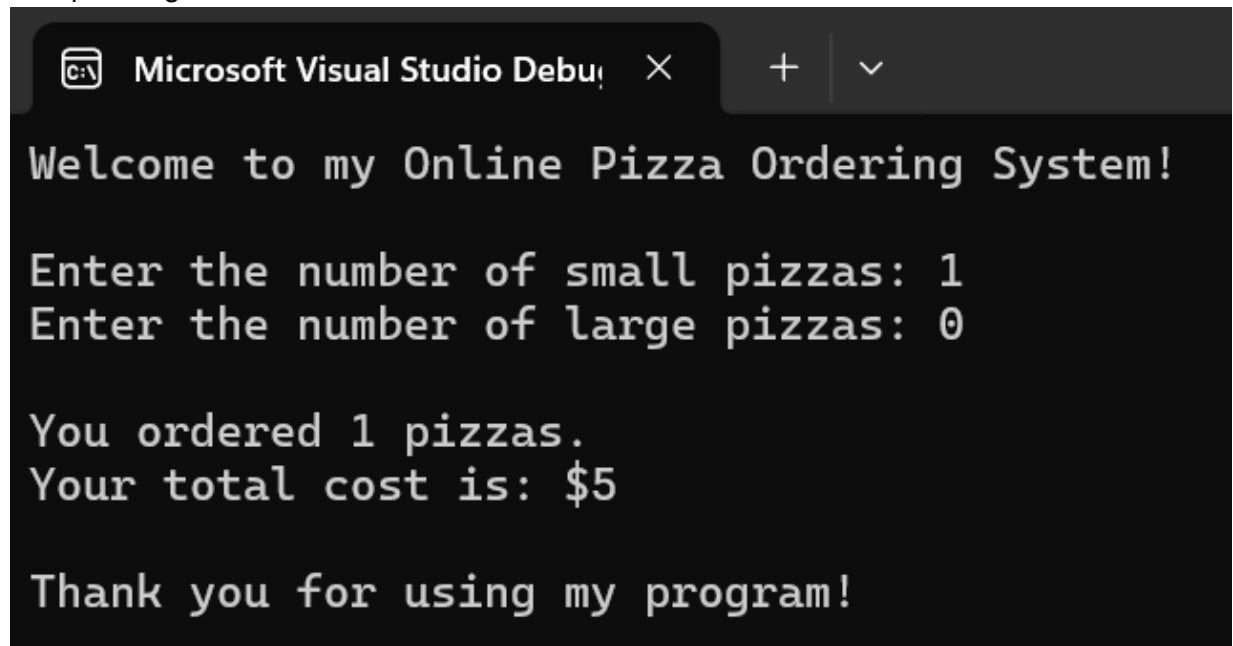DISPLAY "Thank you for using this calculator! Please report any bugs found."

END

e. Include 2 Sample Program Runs for your program using your own set of data. This data set must be different from my Sample Runs in the Assignment document. This process is similar to Unit Testing and will help you test your program better.

Sample Program Run 1:

Sample Program Run 2:

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
		`SET num_dogs = 1`
		`DO`
		`    DISPLAY num_dogs, " dogs!"`
		`    SET num_dogs = num_dogs + 1`

	WHILE *condition*	`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`