CS 161A: Programming and Problem Solving I

Assignment A06 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:

☐ 7. CS 161A: Loops Part II   ■ 50% ■ 84% ■ 100% ⌃

Assigned zyLabs completion screenshot:

	7.7 LAB: Countdown until matching digits	■100%	⌄
☐	7.8 LAB: Count characters	■100%	⌄
☐	7.9 LAB: Count input length without spaces, periods, exclamation p...	■ 0%	⌄
☐	7.10 LAB: Print string in reverse `Optional`	■ 0%	⌄

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 allow you to transact coins for coffee or tea. Simply enter the coins into the input, make your beverage selection, and then this program will display any remaining balance or amount due issues.

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 Coffee/Tea Vending Machine!

Enter coins - 5, 10, or 25 only: 25
Enter coins - 5, 10, or 25 only: 25
Enter coins - 5, 10, or 25 only: 5
Enter coins - 5, 10, or 25 only: 9
Invalid Option! Please choose a valid option!
Enter coins - 5, 10, or 25 only: 10
Enter coins - 5, 10, or 25 only: f
Invalid Option! Please choose a valid option!
Enter coins - 5, 10, or 25 only: 0
``` |

```
Your balance is $0.65

Please pick an option ($0.25 each):
        C/c: Coffee
        T/t: Tea
        Q/q: Quit
t
How many would you like?
1

Your total is $0.25
Your balance is $0.40

Thank you for using my Vending Machine Program!
```

```
Welcome to my Coffee/Tea Vending Machine!

Enter coins - 5, 10, or 25 only: 25
Enter coins - 5, 10, or 25 only: 1
Invalid Option! Please choose a valid option!
Enter coins - 5, 10, or 25 only: 10
Enter coins - 5, 10, or 25 only: 0

Your balance is $0.35

Please pick an option ($0.25 each):
        C/c: Coffee
        T/t: Tea
        Q/q: Quit
t
How many would you like?
2

Your total is $0.50
Your balance is $0.35
Not enough change!! Please add more coins.

Enter coins - 5, 10, or 25 only: 0

Your balance is $0.35

Please pick an option ($0.25 each):
        C/c: Coffee
        T/t: Tea
        Q/q: Quit
q

Your total is $0.00
Your balance is $0.35

Thank you for using my Vending Machine 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.
String userInput Int quantity Int userCoin
b. Identify and list all of the user output and their data types.
Double amountDue Double balance
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 balance = balance + (userCoin * 0.01) SET amountDue = quantity * COFFEE_TEA_PRICE SET balance = balance - amountDue
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.

```
START


DECLARE Const Double COFFEE_TEA = 0.25


DECLARE Double balance = 0.0

DECLARE Double amountDue

DECLARE Int userCoin = 0

DECLARE String userInput = ""

DECLARE Int quantity = -1

DECLARE Int runState = -1


DISPLAY "Welcome to my Coffee/Tea Vending Machine!"


// A hot pile of gargbage spaghetti that would

WHILE runState != 0

- - DO

- - - - DISPLAY "Enter coins - 5, 10, or 25 only: "

- - - - INPUT userCoin

- - - - SET balance = balance + (userCoin * 0.01)

- - WHILE userCoin != 0


DISPLAY "Your balance is ${balance}"


IF runState != 1

- - DISPLAY "Please pick an option ($0.25 each):"
```

```
- - DISPLAY " C/c: Coffee"

- - DISPLAY "  T/t: Tea"

- - DISPLAY "  Q/q: Quit"

- -

- - WHILE userInput != "c" || userInput != "t"

- - - - INPUT userInput

- - - - SET userInput = userInput[0].tolower()

- - - - IF userInput != "c" || userInput != "t"

- - - - - - DISPLAY "Invalid Option! Please choose a valid option!"

- -

- - WHILE quantity >= 0

- - - - DISPLAY "How many would you like?"

- - - - INPUT userInput

- - - - // I am allowing 0 as a psuedo cancel option.

- - - - IF ((int)userInput >= 0)

- - - - - - SET quantity = (int)userInput

- - - - - - SET amountDue = quantity * COFFEE_TEA

- - - - ELSE

- - - - - - DISPLAY "Invalid Option!"

- -

- -

- - IF amountDue > balance

- - - - DISPLAY "Your total is ${amountDue}"

- - - - DISPLAY "Your balance is ${balance}"

- - - - DISPLAY "Not enough change!! Please add more coins."
```

```
- - - - SET runState = 1

- - ELSE

- - - - SET balance = blanace - amountDue

- - - - DISPLAY "Your total is ${amountDue}"

- - - - DISPLAY "Your balance is ${balance}"

- - - - DISPLAY "Thank you for using my Vending Machine Program!"



END
```

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	SELECT *variable or*	`SELECT num_dogs`

| statement | *expression*<br>  CASE *value_1:*<br>    *statement*<br>    *statement*<br>  CASE *value_2:*<br>    *statement*<br>    *statement*<br>  CASE *value_2:*<br>    *statement*<br>    *statement*<br>  DEFAULT:<br>    *statement*<br>    *statement*<br>  END SELECT | ```
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*<br>    *statement*<br>    *statement*<br>  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<br>    *statement*<br>    *statement*<br>  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*<br>    *statement*<br>    *statement*<br>  END FOR | ```
FOR count = 1 TO 10
    DISPLAY num_dogs, " dogs!"
END FOR
``` |

Functions

| Create a function | FUNCTION *return_type name (parameters)*<br>    *statement*<br>    *statement*<br>  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
``` |