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:
N/A

Assigned zyLabs completion screenshot:
N/A

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:

```
# Description:     This program will calculate your total miles cycled
#                                          and tell you if you have met your target goal or not
```

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 Miles Tracker program.


How many miles do you want to ride this week? 156

How many miles did you ride on Sunday? 10

How many miles did you ride on Monday? 60

How many miles did you ride on Tuesday? -4

Miles must be 0 or greater!

How many miles did you ride on Tuesday? -0

How many miles did you ride on Wednesday? 5

How many miles did you ride on Thursday? 22

How many miles did you ride on Friday? 0

How many miles did you ride on Saturday? 23
```

```
You rode 120 miles this week.

Uh oh! You missed your goal by 36 miles!


Keep riding!
```

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).	
# Input: #	Int milesTarget Int milesDay
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).	
# Output: #	Int milesTotal Int goalDif
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.

goalDif = milesTarget - milesTotal;

milesTotal – milesTarget

milesTotal = milesTotal + milesDay;

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.

Copy the below section and paste in your Design Document

I understand that acts of academic dishonesty may be penalized to the full extent

allowed by the Portland Community College Student Conduct Code, including receiving

a failing grade for this exam. I recognize that I am responsible for understanding

the provisions of the PCC Student Conduct Code as they relate to this Course

Challenge Exam.

```
START


DECLARE Function Int Main()
- - DECLARE Int milesTarget
- - DECLARE Int milesDay
- - DECLARE Int milesTotal
- - DECLARE Int goalDif
- - DECLARE runState = 0
- -
- - DISPLAY "Welcome to my Miles Tracker program."
- -
- - CALL inputPrompt(0)
- - DO
- - - - CALL getInput(milesTarget)
- - - - CALL validateInput(milesTarget, runState, TRUE)
- - - - IF runState == 1
- - - - - - CALL consoleClear()
- - - - - - CALL errorHandle(0)
- - WHILE runState != 0
- -
- - IF milesTarget <= 0
- - - - DISPLAY "No miles were tracked this week."
- - ELSE
- - - - milesTotal = CALL calcTotal(milesDay, milesTotal)
- -
- - SET goalDif = milesTarget = milesTotal
- -
- - IF goalDif > 0
- - - - DISPLAY "Uh oh! You missed your goal by {goalDif} miles!"
- - ELSE IF goalDif < 0
- - - - DISPLAY "Great job! You exceeded your goal by {goalDif} miles!"
- - ELSE
- - - - DISPLAY "Good job! You met your goal!"
- -
- - DISPLAY "Keep riding!"



DECLARE Function Void inputPrompt(int messageCode)
- - SWITCH
- - - - CASE(0)
- - - - - - DISPLAY "How many miles do you want to ride this week? "
- - - - CASE(1)
- - - - - - DISPLAY "How many miles did you ride on Sunday? "
- - - - CASE(2)
- - - - - - DISPLAY "How many miles did you ride on Monday? "
- - - - CASE(3)
- - - - - - DISPLAY "How many miles did you ride on Tuesday? "
- - - - CASE(4)
- - - - - - DISPLAY "How many miles did you ride on Wednesday? "
- - - - CASE(5)
- - - - - - DISPLAY "How many miles did you ride on Thursday? "
- - - - CASE(6)
- - - - - - DISPLAY "How many miles did you ride on Friday? "
- - - - CASE(7)
- - - - - - DISPLAY "How many miles did you ride on Saturday? "
- - - - DEFAULT
- - - - - - DISPLAY "Error when calling function inputPrompt with {messageCode}"
```

```
DECLARE Function Void errorHandle(int errCode)

- - SWITCH

- - - - CASE(0)

- - - - - - DISPLAY "Miles must be 0 or greater!"

- - - - DEFAULT

- - - - - - DISPLAY "Error when calling function errorHandle with {errCode]}"



DECLARE Function Int calcTotal(int milesDay, int milesTotal)

- - FOR i = 0; i < 6; ++i - - - - CALL inputPrompt(i + 1)

- - - - DO

- - - - - - CALL getInput(milesDay)

- - - - - - CALL validateInput(milesDay, runState, FALSE)

- - - - - - IF runState == 1

- - - - - - - - CALL consoleClear()

- - - - - - - - CALL errorHandle(0)

- - - - - - ELSE

- - - - - - - - milesTotal = milesTotal + milesDay

- - - - WHILE runState != 0



DECLARE Fucntion Void getInput(int &goal)

- - INPuT goal
```

```
DECLARE Function Void validateInput(int &userInput, int &runState, bool allowNeg)

IF allowNeg = FALSE

- - - - IF userInput < 0

- - - - - - SET runState = 1

- - - - ELSE

- - - - - - SET runState = 0

ELSE

- - runState = 0



DECLARE Function Void clearConsole():

- - CLEAR CIN

- - IGNORE CIN


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 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`