CS 160: Exploring Computer Science
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 it to D2L.

This document contains an interactive checklist. To mark an item as complete, click on the box.

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

- ☑ ~~Step 1: Write a detailed description of your program, at least two complete sentences~~
- ☑ ~~Step 2: If applicable, design a sample run with test input and output~~
- ☑ ~~Step 3: Algorithm design~~
  - ☑ ~~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. Look at the Pseudocode syntax at the bottom of this document. Tools for flowchart - Draw.io - Diagrams.net~~

1. 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 calculate the users BMI or Body Mass Index number. The users BMI will be calculated using the users height and weight. It will also return the BMI category they fall under based on WHO's (World Health Organization) definitions.

2. 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:
What is your weight in pounds/lbs: 120 What is your height in feet (ex. 5ft 4in is 5): 5

What is your height in inches (ex. 5ft 4in is 4): 6

Your BMI: 19.4

You are in the "normal" BMI category according to WHO (World Health Organization)

3. 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, or language syntax)**. Do not include any Python 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. Simple data types include string, integer, floating point, (single) character, and boolean. Complex Data structures like lists should be referenced by name, e.g. "array of integer" or "array of string".
weight: float - The users weight in pounds heightFeet: int - The users height in feet heightInches: float - The users height in inches
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. Complex Data structures like lists should be referenced by name, e.g. "array of integer" or "array of string".
bmi: float - The users BMI number bmiCategory: string - The users bmi category "Underweight, Normal Weight, Overweight, Obesity"
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.If there are no calculations needed, state there are no calculations for

KG = 0.45359237

METER = 0.0254

weight *= KG to convert weight in LBs to KGs

totalInches = (feet * foot) + heightInches

heightMeters = totalInches * METER

bmi = weight / (heightMeters ^ 2)

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 Python or any language specific syntax.**

INPUT weight

INPUT heightFeet

INPUT heightInches

SET weight = weight * 0.45359237

DECLARE float heightMeters

SET heightMeters = (heightFeet * 12) + heightInches

DECLARE float bmi

SET bmi = weight / (heightMeters ^ 2)

DECLARE string bmiCategory

IF bmi <= 18.5 THEN

SET bmiCategory = "Underweight"

ELSEIF bmi >= 18.5 AND bmi <= 24.9 THEN

SET bmiCategory = "Normal weight"

```
ELSEIF bmi >= 25.0 AND bmi <= 29.9 THEN

SET bmiCategory = "Pre-obesity"

ELSEIF bmi >= 30.0 AND bmi <= 34.9 THEN

SET bmiCategory = "Obesity class I"

ELSEIF bmi >= 35.0 AND bmi <= 39.9 THEN

SET bmiCategory = "Obesity class II"

ELSE

SET bmiCategory = "Obesity class III"

END IF

DISPLAY "Your BMI: " + bmi

DISPLAY   "Your BMI Category is " + bmiCategory + " as defined by WHO."
```

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
		`IF num_dogs > 10 THEN`
		`    DISPLAY "You have more than`
		`10 dogs!"`
		`ELSE`
		`    DISPLAY "You have ten or`
		`fewer dogs!"`

	END IF	`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`

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`