Computing I
Programming Assignment 4

In this exercise, you will practice the top-down approach to program design and implementation. Recall that, with this approach, we assume that any other functions on which a particular function depends operate as expected. We use stub functions as temporary placeholders when designing and implementing the program, and as we iterate, we refine these stubs by writing the C code that implements the required functionality.

program4a.c: Converting lengths and weights

Write a program that asks the user if he or she wants to convert values that are lengths or weights. If the user chooses lengths, the program calls a stub function `convert_lengths` that simply indicates the user's choice. If the user chooses weights, the program calls a stub function `convert_weights` that simply indicates the user's choice. Use the value 1 to indicate lengths and the value 2 to indicate weights. If the user enters 0, terminate the program. The program should continue to prompt the user for input until he or she correctly enters a value of 0, 1, or 2. In addition, the program should allow the user to call `convert_lengths` or `convert_weights` as many times as desired (i.e., until a value of 0 is input, indicating that the user is finished). Notice that this style of loop does not require you to ask the user if they want to continue but instead assumes they want to continue until they select 0.

program4b.c: First iteration

Make a new project and copy the code from part 4a into this project to begin. Design and implement the stub functions `convert_lengths` and `convert_weights` from Program 4a.

The function `convert_lengths` will ask the user if he or she wants to convert lengths from feet/inches to meters/centimeters (input value of 1) or from meters/centimeters to feet/inches (input value of 2). An input value of 0 indicates that the user no longer wants to convert length measurements. Based on the user's input, the `convert_lengths` function will call a stub function `length_to_metric` for an input value of 1, a stub function `length_to_us` for an input value of 2, and return control to the main program for an input value of 0. Returning control to the main program simply requires your function to return. You should not ever call the function main from your program. The `convert_lengths` function continues to prompt the user for input until he or she correctly enters a value of 0, 1, or 2. The new stub functions should simply indicate the user's choice by printing an appropriate message.

The function `convert_weights` will ask the user if he or she wants to convert weights from pounds/ounces to kilograms/grams (input value of 1) or from kilograms/grams to pounds/ounces. An input value of 0 indicates that the user no longer

wants to convert weight measurements. Based on the user's input, the `convert_weights` function will call a stub function `weight_to_metric` for an input value of 1, a stub function `weight_to_us` for an input value of 2, and return control to the main program for an input value of 0. The `convert_weights` function continues to prompt the user for input until he or she correctly enters a value of 0, 1, or 2. The new stub functions should simply indicate the user's choice by printing an appropriate message.

program4c.c: Second iteration

Make a new project and copy the code from part 4b into this project to begin. Design and implement the stub functions `length_to_metric`, `length_to_us`, `weight_to_metric`, and `weight_to_us` from Program 4b.

The function `length_to_metric` should read the length in feet and inches and output the equivalent length in meters and centimeters. To accomplish this task, `length_to_metric` should use three additional functions: one to input the values, one to do the conversion, and one to output the results. Similarly, the function `length_to_us` should read the length in meters and centimeters and output the equivalent length in feet and inches. To accomplish this task, `length_to_us` should use three additional functions: one to input the values, one to do the conversion, and one to output the results. (Note: there are 0.3048 meters in a foot, 12 inches in a foot, and 100 centimeters in a meter. You should not look up other conversion factors online. Use these.)

The function `weight_to_metric` should read the weight in pounds and ounces and output the equivalent length in kilograms and grams. To accomplish this task, `weight_to_metric` should use three additional functions: one to input the values, one to do the conversion, and one to output the results. Similarly, the function `weight_to_us` should read the weight in kilograms and grams and output the equivalent weight in pounds and ounces. To accomplish this task, `weight_to_us` should use three additional functions: one to input the values, one to do the conversion, and one to output the results. (Note: there are 2.2046 pounds in a kilogram, 1000 grams in a kilogram, and 16 ounces in a pound. You should not look up other conversion factors online. Use these.)

Please note that length in feet and inches is a single unit of length. If I say that I am 5 feet 9 inches tall then that means that I am describing one length using 2 units of measure. The larger unit, feet in this case, should be an integer in your programs and the smaller unit, inches, should be modeled using a double. The same should be true for weight measurements in that they describe a single weight but use 2 units of measure. To be clear, this assignment is not asking you to convert feet to meters and then convert inches to centimeters. It is asking you to take one length given in some number of feet and some number of inches and convert that length to an equivalent measure using both

meters and centimeters. As an example, the length: 5 feet and 10.5 inches is equivalent to 1 meter and 79.07 centimeters.

Command for submitting:
submit dbadams program4 program4a.c program4b.c program4c.c

Be sure to submit the correct files the first time. All programs must have a section of comments at the beginning that give the program number, your name, the date, the time you spent working on the program, and its purpose. You will be graded on the correctness of your program, the comments, and the general layout of your code (indentation, etc.) according to the following scale:

Correctness – 70 %
Comments – 10%
General layout – 20%

A program that does not compile or link will not be graded.

A sample comment section is given below.

```
/*
  Program:     <name of program>
  Author:      <your name>
  Date:        <date you finish the program>
  Time spent:  <total amount of time spent on the project>
  Purpose:     The purpose of this program is to
               demonstrate an acceptable comment section
               for my program.  My code does not actually
               do anything.
*/

int main(int argc, char* argv[])
{
  // Insert your code here.

  return 0;
}
```