Moon Weight

When I first started coding, I remember not knowing where to begin when presented with a problem statement. This guide is intended to help you get started by breaking down a problem statement into parts before converting it to pseudocode. This problem should be familiar—it is the first challenge question from the CSE 1321 Module 1 slides.

As the semester progresses, try working through other challenge questions on your own—first in PSEUDOCODE, then in your lab language. Don't be shy about using the Pseudocode Guide as you go. If you get stuck, swing by the Tutoring Lab.

Happy coding!

Professor Kendrick

Problem Statement

Write a program that asks the user's weight (in pounds) and prints out how much s/he weighs on the moon.

NOTE: Your moon weight is 16.5% of your Earth weight.

STEP 0: Use the problem statement to identify the things we'll need

All computers do four things: input (what we put in), memory (storing information), processing (following the instructions we give it), and output (displaying info out). We can pull this information out of the problem statement to help us find the parts we need for our PSEUDOCODE.

Input - ask the user's weight in pounds

Memory - store the user's Earth weight and their moon weight.

Processing - convert user's Earth weight to their moon weight—which is 16.5% of their Earth weight.

Output - display a message to the user showing the results of our calculations (their moon weight).
STEP 1: Start with MAIN

We know that any program needs a MAIN (the skeleton), and in PSEUDOCODE, it looks like this:

```
MAIN
BEGIN
    //stuff goes here
END
```

The two slashes represent a COMMENT, documentation that programmers include to provide information about their code. When you write in your lab language, any line starting with two slashes is a comment and the compiler will ignore everything on the line with those symbols. Also notice that the comment is indented, which represents that it is “inside” of the BEGIN and END.

Here's what this looks like in each of your lab languages. We call it the SKELETON--the smallest program you can write. You don't need to understand every part of this in the syntax of your language (yet), you simply need to know that all the parts are necessary for your program to run, and that your stuff goes where the comment is. See the links below to see the skeletons in each language.

Java  |  C#  |  C++

STEP 2: Declaring Variables

In this case, the problem statement tells us that we're asking the user for their weight (in pounds). We'll need to store that information in a VARIABLE, a name we'll use to reference a data storage location. There are two steps to use for VARIABLES: declaration and initialization or assignment. When we declare a VARIABLE, we give it a name (and, if we're working in the syntax of our lab language, assign its data type). You can choose to initialize the variable with a particular value (i.e. `userWeight = 160`) or you can wait to assign a value later in the program (typical when reading user input). We'll also need a VARIABLE to hold the new weight we'll calculate. In PSEUDOCODE, declaring (creating) VARIABLES looks like the code below. Note: we did NOT give the variables a starting value; we'll do that later.

```
MAIN
BEGIN
    CREATE earthWeight
    CREATE moonWeight
END
```

We talked in class about using VARIABLE names that describe the information that they'll contain. I chose `earthWeight` and `moonWeight`, and I used camelCasing instead of underscores. I could have used `earth_weight` and `moon_weight` and that would be fine, too. The important thing is that it is evident what is housed in each VARIABLE.
STEP 3: Prompting the User

We also need to prompt the user to enter their weight, then store their information inside of the earthWeight VARIABLE. In PSEUDOCODE, we'll use the PRINT or PRINTLINE keyword to indicate that we're printing a message to the console. I like to use PRINT when requesting information so that the answer the user inputs sits on the same line. Whatever you choose to do, you should have a good reason for doing so and be consistent throughout your program. Here's the PSEUDOCODE to prompt the user:

```
MAIN
BEGIN
    CREATE earthWeight
    CREATE moonWeight
    PRINT "Please enter your weight in pounds: "
END
```

Though it’s not required, don't forget to include a space after your colon if you're going to use a PRINT so that their response isn't right up against your prompt.

STEP 4: Storing user input

Once you prompt the user, storing their information is a simple matter of assigning their value to the variable that it will be housed in. We READ in the information, then ASSIGN it to the variable. In syntax, we'd use the ASSIGNMENT OPERATOR (=). In PSEUDOCODE, we use a left facing arrow to show that we're storing the value we read inside of the variable we declared earlier in the program. Instructions for the left facing arrow are in the Module 2 Pseudocode Guide. Let's update our PSEUDOCODE:

```
MAIN
BEGIN
    CREATE earthWeight
    CREATE moonWeight
    PRINT "Please enter your weight in pounds: "

    earthWeight ← READ user input

END
```

NOTE: In most actual languages, you may have to do some work to convert the user's input from STRING to whatever data type is needed to match the variable you're storing the data in. We don't worry with that in PSEUDOCODE, but you will when you convert the problem into the syntax of your lab language. Also, note that it is OK to put “earthWeight ← READ” as a shorthand.
STEP 5: Do the math

Now that we have earthWeight, we can calculate the moonWeight for the user. The problem statement tells us that the moonWeight is 16.5% of earthWeight. We'll do that calculation and assign it to the moonWeight variable. Here it is in PSEUDO CODE:

MAIN
BEGIN
CREATE earthWeight
CREATE moonWeight
PRINT "Please enter your weight in pounds: "
earthWeight ← READ user input
moonWeight ← earthWeight * 0.165
END

STEP 6: Print the results

All that's left to do now is print the results out so the user knows their moon weight. I'm going to use a PRINTLINE instead of a PRINT here (because I'm not sure what I might add later, and whatever it is I know I don't want to display it on the same line as this message). I also have to CONCATENATE (or “add”) the variable moonWeight and the STRING message we're displaying to the user.

MAIN
BEGIN
CREATE earthWeight
CREATE moonWeight
PRINT "Please enter your weight in pounds: "
earthWeight ← READ user input
moonWeight ← earthWeight * 0.165
PRINTLINE "On the moon, you would weigh " + moonWeight + " lbs."
END

Notice that I've included spaces in the strings on either side of the VARIABLE, and that I've used + to denote CONCATENATION. Some languages use different syntax for this, but + is fine to use in PSEUDO CODE.