Pseudocode Step-By-Step

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 one of the challenge questions 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 K

Challenge Question: Moon Weight

Write a program that asks a 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 identify the things we'll 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 the user's Earth weight to their moon weight (16.5% of their Earth weight).

Output – display a message to the user with the results of our calculations (their moon weight).

STEP 1: Starting with MAIN

The MAIN is the "entry point" or "starting point" of the program. All programs have a main. In pseudocode, the MAIN looks like this:

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

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 MAIN.

Here's what this looks like in each of your lab languages. We call it the skeleton: the bare-bones code you need to make your program compile and 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.

Click 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/assignment. When we declare a variable, we give it a name (and, if we are working in the syntax of our lab language, assign its data type). You can choose to initialize the variable with a 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: in this example, we did NOT give the variables a starting value. We'll do that later.

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

We talked in class about using variable names that describe the information they contain. I chose earthWeight and moonWeight and 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 from the name what will be 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 as the prompt. 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
     CREATE earthWeight
     CREATE moonWeight

     PRINT "Please enter your weight in pounds: "
END MAIN
```

Don't forget to include a space after your colon if you're using a PRINT so that the user's response isn't displayed right up against the 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, storing it in the earthWeight variable.

Let's update our pseudocode:

```
MAIN
     CREATE earthWeight
     CREATE moonWeight

     PRINT "Please enter your weight in pounds: "
     READ earthWeight
END MAIN
```

NOTE: In some programming languages, you'll have to do some work to convert the user's input from string to whatever data type is needed so that it matches the variable you're storing the data in. We don't worry about that in pseudocode, but you will need to think about it when you convert the problem into the syntax of your lab language.

STEP 5: Performing Calculations

Now that we have the earthWeight, we can calculate the moonWeight for the user. The problem statement tells us that someone's moonWeight is 16.5% of their earthWeight. We'll perform that calculation and assign the results to the moonWeight variable.

Here it is in pseudocode:

```
MAIN
     CREATE earthWeight
     CREATE moonWeight

     PRINT "Please enter your weight in pounds: "
     READ earthWeight
     moonWeight = earthWeight * 0.165
END MAIN
```

STEP 6: Printing the Results

All that is left to do now is to display the results back to the user, so they know their moon weight. I'm going to use a PRINTLINE instead of a PRINT here (because I'm not sure what I might add to the program later, but whatever it is I know I won't want to display it on the same line as this message). I will also need to concatenate (or join) the variable moonWeight and the string message we're going to display to the user.

```
MAIN
     CREATE earthWeight
     CREATE moonWeight

     PRINT "Please enter your weight in pounds: "
     READ earthWeight
     moonWeight = earthWeight * 0.165

     PRINTLINE "On the moon, you would weigh " +
     moonWeight + "lbs."
END MAIN
```

Notice that I've included spaces in the strings on either side of the variable, and that I've used + to denote concatenation. Concatenation means "to join together." The concatenation operator is used to join strings (and other things) together. Some languages use different syntax for this, but + is fine to use in pseudocode.