# ­­­Description

The purpose of this Lab is to reinforce your understanding of the material in Topic 2. In the activity section of this lab, you will experience "the design process." A great deal of time is spent by the higher-paid systems architects and engineers designing and documenting an application before it gets to the programmers for coding. Programmers, too, must design and document code based on the requirements of the program. A major step in the process is going from requirement to algorithm to pseudo code. Well written pseudocode translates well to most programming languages.

# Part 1: Questions

Complete the assignment in Topic 2 named, "Lab 2 Questions", in Canvas.

# Part 2: Activity

## Definitions and Descriptions

### Algorithms

An algorithm is a set of well-designed logical steps that must take place in order to solve a problem. The flow the algorithm is sequential. For example, before you process calculations, all data needed should be retrieved.

### Pseudocode

Pseudocode is an informal language that has no syntax rules and is not meant to be compiled or executed. The flow the program is sequential. For example, before you ask for input, you should display what information you want from the user.

// Comments are typically done by putting two forward slashes before the lines you

// want to document. Comments are used to explain code.

Variables are named storage locations. "Declare" is the word used before naming a variable. Data types are: Real for decimal numbers, Integer for whole numbers, and String for a series of characters. Follow the rules for naming variables:

1. Must be one word with no spaces.
2. Usually no punctuation characters, only letters and numbers.
3. Name cannot start with a number.

"Display" is the word used to print something to the screen. Any information needed to be displayed to the user should be put inside quotation marks such as Display "This is how you print something to the screen”. When using display to print both a string and the value of a variable, a comma is used, such as Display "Here is the average”, average.

"Input" is the word used to get the user to enter data. The data value entered by the user will be placed in the variable that follows the keyword input such as Input variableName.

"Set" is the word used before a calculation. Standard math operators are used, such as + - \* / MOD ^. Operators can be combined in one calculation, but it is wise to group expressions together using parentheses. Remember the order of operations. Some examples are Set sale = price – discount and Set average = (test1 + test2 + test3) / 3.

**\*\*\* See Appendix C in the book for more information.**

### C++ Code

Line comments in C++ are preceded by //. Block comments in C++ are preceded by /\* and they are ended with \*/.

Notice that most statements in C++ are terminated by a semicolon (;). This is a requirement.

Input of strings into a variable can be done using the *cin* operation. This operation converts the input to a series of characters so they can be used later in the program. However, this operation can only input one word and we will stick to this limitation as this time. This is often written as an equation such as   
 std::cin >> *stringVariabl*e*;*

Input of numeric values into a variable is done using the *cin* operation as well. The method of input is the same as string input. For example,   
 std::cin >> *realVariable;*

Equations are written similarly to the method used in pseudocode, but without the *Set* keyword. For example,   
 *total = apples + oranges;*

Complex formulas should use parentheses to group processes. In addition, if input values are taken in as integers, but will be used to calculate a decimal value, they must be converted to real values as needed. For example,   
 *average = (test1 + test2) / 2;*

To display information to the screen, the *cout* operation is used with the string, which is written within double quotation marks. If the value of a variable needs to display after the string, use another set of <<. For example,   
 std::*cout << "The average is” << average;*

### Use the following program requirements to complete the lab.

Write a program that will take in basic information from a student, including student first name, abbreviation of degree name (like CSCI), number of credits taken so far, and the total number of credits required in the degree program. The program will then calculate how many credits are needed to graduate. Display should include the student name, the degree name, and credits left to graduate.

## Part 1

Examine the following algorithm.

1. Get the student first name.
2. Get the abbreviation of degree name.
3. Subtract the number of credits taken so far from the required credits for the degree.
4. Get the number of credits required for the degree program.
5. Get the number of credits the student has taken so far.
6. Display the input information in Step 1 and 2.
7. Display the calculated information.

Answer the following:

1. Analyze the algorithm. Are there any logic errors? If so, describe how you would fix them. Refer to line numbers when possible. Yes there are. I would fix them by moving line 3 to being after line 5. This would make line 4 the new line 3 and line 5 the new line 4.
2. What steps require user interaction (where the user must enter some input)?  
   Steps 1,2,4, and 5.

## Part 2

1. This program is most easily solved using only five variables. Identify potential problems with the following variables declared in the pseudocode. Assume that the college can **offer half credits like 3.5**.

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Problem? (Y or N)** | **If Yes, what's wrong?** |
| Declare Real creditsTaken | N |  |
| Declare Real credits Degree | N |  |
| Declare Integer creditsLeft | Y | The amount of credits left may be a decimal, or a half credit |
| Declare Real studentName | Y | Names should be stored under a string. |
| Declare String degreeName | N |  |

1. Complete the pseudocode using the corrected variables from above.

// variable declarations

Declare Real creditsTaken

Declare Real creditsDegree

Declare Real creditsLeft

Declare String studentName

Declare String degreeName

// Prompt for and take user input

// Example of a prompt: Display "Enter student's first name: ”

Display “Enter student’s first name: “

Input studentName

Display “Enter the degree name: “

Input degreeName

Display “How many credits does this degree need?”

Input creditsDegree

Display “How many credits has the student taken?”

Input creditsTaken

// Calculate the remaining credits

Set creditsLeft = creditsDegree - creditsTaken

// Display student name, degree program, and credits remaining.

Display “The student’s name is “, studentName,“they are earning a”, creditsDegree,“ degree, and have ”, creditsLeft“ credits left.”

1. Write the exact output you would expect from the following line of code if the user of the program enters "Bill”. Enter it in the following textbox.  
   Display "The student's first name is ", studentName

The student’s first name is Bill

1. Write the exact output you would expect from the following line of code if the user of the program enters a degree that is 63 credits in total and they have taken 40 credits.

This program requires 63 credits and they have taken 40 credits so far.

Display "This program requires ", creditsDegree, " credits and they have taken ", creditsTaken, " so far.”

## Part 3

Translate your pseudocode to actual code using C++.

**See Canvas for the template C++ file for this part of the lab. Rename the template to, "student\_credits.cpp"**

1. Your code’s output should output the following given as input the information in bold followed by and “enter” (<Enter>) as follows:

Enter student's first name: **Jane<Enter>**

Enter degree name: **CSCI<Enter>**

Enter the number of credits required: **63<Enter>**

Enter the number of credits taken so far: **24<Enter>**

Student's first name is Jane.

The degree name is CSCI.

There are 39 credits left until graduation.

1. Describe your experience and lessons learned with this lab.  What is the most challenging aspect of this lab?

It was an overall simple lab. Very straightforward in its instructions and steps. The most challenging aspect was probably figuring out that the degree name and student name had to be strings and knowing how to put that into your code.

# What to Submit

1. Answer the questions in this file. Change its name to, "<username>\_lab\_2.docx" where *<username>* is replaced with your Mt SAC username.
2. Make sure your program runs and its output is formatted properly.
3. Submit your completed lab work (this document) and the C++ source code file named "student\_credits.cpp".