**Assignment #1**

**Computer Programming**

**Deadline Oct. 28, 2022 6:00 PM**

Discuss the following in details. Please note:

* Your submission should have a front-page with Assignment title, Course Name Section, Student name, Date of Submission.
* Font Times New Roman 12
* Reference your discussion with **APA formatting**

1. Create a reflection paper about Fundamentals of Programming. Discuss the importance and relation of Programming to ECE students.

* 1 page for reflection paper and 1 front-page in APA 7th format.

# [*Fundamentals of Programming*](http://ece.uwaterloo.ca/~ece150/)

“Many of you are here because you are in electrical engineering. You find yourself in a course where you are learning to "program", which as far as you're concerned has nothing to do with what you're actually interested in. Indeed, perhaps you picked electrical engineering specifically because you are not interestd in programming.

After all, you want to build *things* to solve problems and to apply the principles of electrical engineering (essentially, applying Maxwell's equations through the lens of calculus to control the emission and flow electrons and photons) to real-world problems to find solutions that will benefit your employer, your client, and society in general.

Instead, you're here learning about how to calculate Fibonacci numbers (which you can already do), how to search through a sorted list (what you learned to do the first time you needed a telephone book or an index), or or how to sort a list of things from smallest to largest (what you learned to do with the Fisher-Price [ring stacker](https://www.fisher-price.com/en_CA/products/Rock-a-Stack)). You've probably already solved the problem of classes because you've created index cards that contains, for example, people's names, birthdays, phone numbers, etc. You could sort those cards by the first names, by the last names, or by their birthdays.

To start, it's a question of efficiency: computers can organize and process data much more quickly than a human can, and you will come across situations where you will be required to automate repetitive tasks, and the tools you learn in programming will help you achieve the automation elsewhere. When you preform the above operations, you are using a massively parallel neural network called your brain. In this course and your algorithms and data structures course, you will learn how to perform such tasks efficiently, to minimize cost and the likelihood of error. Over time, you will also learn that many complex problems come down to solving numerous smaller and simpler problems, but the only way to achieve this in a reasonable amount of time is to improve the efficiency.

Also, learning a programming language is the first step to learning rigorous logical, structured and ordered thinking. When you design circuits, you will also be using a language, but it will be a language like VHDL (for VHSIC Hardware Design Language where VHSIC stands for Very High Speed Integrated Circuit):

-- fpga4student.com: FPGA projects, Verilog projects, VHDL projects

-- VHDL project: VHDL code for a comparator

-- A comparator with 2 2-bit input A and B, and three outputs

-- A less than B, A equal B, and A greater than B

-- The comparator is designed by using truth table, K-Map.

-- The final output expressions are used for VHDL coding

library IEEE;

use IEEE.std\_logic\_1164.all;

entity comparator\_VHDL is

port (

A,B: in std\_logic\_vector(1 downto 0); -- two inputs for comparison

A\_less\_B: out std\_logic; -- '1' if A < B else '0'

A\_equal\_B: out std\_logic; -- '1' if A = B else '0'

A\_greater\_B: out std\_logic -- '1' if A > B else '0'

);

end comparator\_VHDL;

architecture comparator\_structural of comparator\_VHDL is

signal tmp1,tmp2,tmp3,tmp4,tmp5, tmp6, tmp7, tmp8: std\_logic;

-- temporary signals

begin

-- A\_equal\_B combinational logic circuit

tmp1 <= A(1) xnor B(1);

tmp2 <= A(0) xnor B(0);

A\_equal\_B <= tmp1 and tmp2;

-- A\_less\_B combinational logic circuit

tmp3 <= (not A(0)) and (not A(1)) and B(0);

tmp4 <= (not A(1)) and B(1);

tmp5 <= (not A(0)) and B(1) and B(0);

A\_less\_B <= tmp3 or tmp4 or tmp5;

-- A\_greater\_B combinational logic circuit

tmp6 <= (not B(0)) and (not B(1)) and A(0);

tmp7 <= (not B(1)) and A(1);

tmp8 <= (not B(0)) and A(1) and A(0);

A\_greater\_B <= tmp6 or tmp7 or tmp8;

end comparator\_structural;

This code defines a circuit that takes two input voltages V1V1 and V2V2, and has three outputs, where the three outputs are high if V1<V2V1<V2, V1=V2V1=V2, and V1>V2V1>V2, respectively, and low otherwise. It is available at the website [FPGA 4 Student](https://www.fpga4student.com/2017/08/vhdl-code-for-comparator.html).

Looking through this, you will see many commands that are similar to those used in C++; thus, learning the mechanisms for programming in C++ will help you learn to use VHDL.

Additionally, to learn a programming language, you will learn to understand binary, and binary is in a sense the behavior of a [transistor](https://en.wikipedia.org/wiki/Transistor), and transistors are ubiquitous within the field of electrical engineering.

Additionally, you will be running simulations of your hardware. It is possible to build your device and then test it, but given the design of the device, you can also simulate the behavior using a programming language such as C++ or MATLAB, both of which are high-level programming languages. In some cases, you may be collating data in a spreadsheet, and the effective manipulation of data in spreadsheets requires programming knowledge. The language may be different (usually Visual Basic for Applications or VBA), but knowing one language makes learning a second language much easier.

In many occasions this author has read on various posts that claim there are still electrical engineering careers that do not require programming knowledge, but those are becoming more and more rare. In general, the ability to program has great advantages and is considered an asset. Remember, programming is a skill, not a profession.

To emphasize that it is a skill, remember that there are some elementary-school students learning to program, nearly half a decade before they get to post-secondary studies. You do not, in general, see elementary-school students learning calculus or electronic circuits, although there are kits you can buy that can facilitate learning elementary circuits, too. For those who do not spend time to cultivate their ability to program, the exactness that is required is often the most intimidating aspect. One semi-colon out of place can be the difference between a perfectly functioning program and one that will not even compile. At the same time, circuits require equally precise timing in their designs, so the need for rigor will not go away simply because you chose to ignore your introductory programming course.

As an interesting aside: some students do not believe that they can learn to program. Here is an interesting video that describes, at a high level, the difference between fixed versus growth mindsets:

[Fixed Mindset versus Growth Mindset](https://www.youtube.com/watch?v=Xv2ar6AKvGc).

Programming is a skill that can be learned just like speaking a second language. Don't expect yourself to be able to solve the world's problems (like a self-driving car) overnight. You don't learn he subtleties of, for example, the [pluperfect verb tense](https://en.wikipedia.org/wiki/Pluperfect) when first learning a new language. One nice feature that programming languages have that spoken languages lack is simplicity.

Additionally, many electrical engineers will work with systems that are closely related to their fields; for example, many electrical systems will be intimately connected with embedded hardware systems. These are microprocessor-controlled systems that are often programmed in C or C++.

Because you will be working with other engineers on a project, while you may not have expertise in software engineering or computer engineering, having familiarity with systems related to electrical engineering will vastly enhance communications.

Some of you may be focused on moving into power engineering, and this has been to this point one of the last bastions of pure-electrical engineering; however, even today many power systems are integrating [smart grids](https://en.wikipedia.org/wiki/Smart_grid).

Now, most of these comments have been *carrots*, meaning that we're trying to demonstrate to you why gaining competence in a programming language is critical to your success as an electrical engineer; however, the following is more of a *stick*: most of the entry-level co-op placements will be in software- or programming-related positions. You will be competing with computer, mechatronics and software engineering students for these entry-level co-op placements. Landing a developer position will be significantly more beneficial to getting a good second placement than landing a position in technical support or quality assurance.

Additionally, electrical engineering students will have to take both ECE 250 *Algorithms and data structures* and ECE 222 *Digital Computers* in your third academic term. Gaining competence in programming in your first term will make these terms much easier, especially as you would likely prefer to focus on ECE 240 *Electronic circuits 1*.”