Hardware vs. Software, and more!

Before we begin discussing some fundamental computer terms, I want to take a step back (no, I didn't say I have to "stumble" back. I know what you're thinking, that I've had too many of those 4 oz, 2 oz gins). Well, no. What I really want to do is define some very basic concepts, to ensure that we are all on the same page.

First of all, I want to define the difference between a "Computer Programmer" and a "Computer User".

Today, lots of computer literate people know how to *use* a variety of computer programs at work and home (such as word processors, spreadsheets, graphics packages, and games). People who use these programs are considered computer *users*. People who *write* (or create) these programs are called *computer programmers*. Knowing how to use the computer is not the same as knowing how to communicate with the computer. An analogy: Just because a person is able to drive a car, does not make him/her an expert mechanic! We have to know how to communicate with the computer, in its own language, before we can become a computer programmer. (Hence, the reason you are all here taking this course!)

Next, I would like to define what a computer actually is. I know most of you already know (I saw a few of those eyeballs rolling), but again, I want to be sure that we are all on the same page!

There are many definitions of a computer. One definition is: A computer is an electronic device capable of performing computations at speeds millions/billions of times faster than humans can.

For example, a computer can perform millions of additions per second. It would takeÂ us humans decades, even using a calculator to do the same job!Â Every job that a computer is required to do is broken down into very simple instructions (add, subtract, test if zero, etc.) We sometimes measure the speed of a computer in MIPS -- which stands for "Millions of Instructions per Second". Soon we will be going to "BIPS" -- Yes. You guessed it. "Billions of Instructions per Second." Amazing, isn't it?! (My kids can't even complete a single instruction from me all day! No wonder I like computers so much!)

To solve a problem using a computer, we as programmers, must express the solution to the problem by using the instruction set for a particular computer language. The C programming language has a wonderful instruction set, consisting of hundreds of basic operations, as well as more complex functions (discussed later) available to the programmer to help solve the problem. When we write a program, the computer will do exactly what we have instructed it to do. Nothing more, nothing less.

So, basically, a **computer program** is actually just a **collection of the instructions necessary to solve a specific problem.**

A good practice before attempting to solve any problem is to first think about the problem and the solution. Then come up with a plan. A design. In computer programming, this design is called an "**Algorithm**". An Algorithm is **an approach or method for solving the problem**. Even if you simply jot down a few lines on paper, it is so important that you think about the solution before attempting to code it up! For example, if I gave you the following problem: Write a program which prompts the user for a number (input), and determines if the number is odd or even. Then display the results (output). A QUICK algorithm might look something like:

1.0 Prompt user for a number.

2.0 Determine if number is odd or even.

2.1 Divide number by two.
2.2 If remainder is zero then
number is even.
Otherwise
number is odd.

3.0 Display results.

Now, we could attempt to write some C code (using its instruction set) which follows the above logic. If you have any problems with your "logic", you will most likely see the problem in your algorithm. It is a lot easier to fix a "logic" problem during design, then once you have coded up your solution. Your algorithms can be as detailed or as high-level as you desire for this class, however in the "real-world" you may be asked to create algorithms using software tools, or you may be asked to write more detailed algorithms (called pseudo-code), or perhaps even flowcharts!!

In case you're wondering, for this class I do not require you to submit algorithms along with your programming assignment. I am going to assume that you are writing your algorithms before you code. Remember, taking the time to design will save you loads of time when you test your program. Without a good design, you code could be riddled with design flaws, giving you logic errors. It will be very difficult to fix the problem at this point, during your coding phase. It would have been better to find the logic flaw during

your design phase. (Have I stressed this enough yet?!?!?)

Now, we can finally, get to the real reason we are in this particular section of lecture notes: To define the difference between Software and Hardware.

Hardware is **the actual machine (system unit) and its supporting devices**. Examples of hardware items are listed below:

- Keyboard (I)
- Mouse (I)
- Printer (O)
- All-in-one Printer (I/O)
- Monitor (O)
- Touchscreen Monitor (I/O)
- CPU (P)
- Memory chips (S)
- Scanner (I)
- System bus (P)
- Disks (S)
- Microphone (I)
- Speakers (O)
- etc.

Legend:

(I) = **Input device**: Data is brought into the computer via this type of device.
(O) = **Output device**: Data is sent out from the computer via this type of device.
(I/O) = **Input and Output device**: Data can be both input to and output from this device.
(P) = **Processing device**: Data is processed using this type of device.
(S) = **Storage device**: Data is stored in this type of device.

Software is **the set of instructions that make the computer do something**.
Examples of software items include:

- Word Processors (MS-Word, Corel Wordperfect, Lotus AmiPro)
- Spreadsheets (MS-Excel, Lotus 123)
- Databases (MS-Access, Borland's dBASE)
- Games (Solitaire, Mind Sweep)
- Operating Systems (DOS, Windows, UNIX, Mac OS X)
- Compilers (cc, Borland's Turbo C++, MS-Visual C++)
- etc.

The basic rule of thumb: If you can touch (break) it, it is hardware!