Assignment 1

1. [9 pts] Define the von Neumann’s architecture. Include both how a von Neumann machine is wired and the process used to execute programs.

The von Neuman Architecture:

* Computers permanently hardwired with a small set of general-purpose operations
* Data and programs are stored in memory
* Fetch-Decode-Execute Cycle

1. [9 pts] Define the von Neumann Bottleneck and explain how it is related to the Processor-Memory performance gap.

Instructions and data are transmitted back and forth between memory and CPU. CPUs generally complete operations faster than memory does. This creates a bottleneck, where CPUs are idle/waiting much longer than the memory is. CPUs have become exponentially faster than memory due to technological advancements over time.

1. [9 pts] List the four computational paradigms discussed *in class*.

**Imperative Languages:** Language using sequential statements, variables, and variable assignment

**Functional Languages:** Treats programs as functions, treats functions as data, and avoids mutable data

**Logic Programming:** Based on formal symbolic logic

**Object Oriented Languages:** Allows programs to write reusable code

1. [9 pts] How are compilers and interpreters similar? How are they different?

* Compilation:
  + Programs are translated into machine language (slow translation/fast execution)
* Interpretation:
  + No translation **beforehand**, slower execution (10 to 100 times slower)

1. [9 pts] What two language design criteria focus areas would you prioritize when designing a programming language? Why?

I would prioritize **Efficiency** and **Regularity**. Coming from a developer’s perspective, the language should work in a way that makes sense to me, as well as anything in general. To me, efficiency seems to be the focus area that most directly affects the programmers daily experience as well as productivity, which is what matters most to me. A language that prioritizes regularity would also reduce frustration while programming (principle of **least astonishment**).

1. [9 pts] Discuss two or more examples of efficiency in the Java programming language. Specify if your example is computational efficiency or programmer efficiency.

**Writability (Programmer Efficiency):** Methods called on an object/class are directly attached to the object

* Object.method(Type param) , Math.random() , etc.

**Readability (Programmer Efficiency):** Most operators in Java are standard to popular programmer languages, making it easier to understand what is happening

* + (concat / increment), - (decrement) , = (assign) , ( == ) compare, etc.

1. [9 pts] Discuss two or more examples of inefficiency in the Java programming language. Specify if your example is computational efficiency or programmer efficiency.

**Writability (Programmer Inefficiency):**

* Using standard IO requires the import of a class (java.util.Scanner / java.io.File), should be built in base-level Java
* Simply printing to stdout requires lots of typing
  + Java: System.out.println(“foo”);
  + Ruby: puts “foo”
* A semi-colon is required after every line of code

1. [9 pts] Discuss two or more examples of regularity in the Java programming language.

**Uniformity:** Conditional statements/blocks look/are typed very similar

* if ( condition ) { code block }
* while ( condition ) { code block }

**Generality:** Most class instances are instantiated similarly

* Type/Class varName = [class specific];

1. [9 pts] Discuss two or more examples of irregularity in the Java programming language.

**Lack of Orthogonality:** primitive types use value while object types use reference semantics

**Lack of Generality:** Using ‘==’ on a primitive type is different from its use on an object. To achieve the same effect with objects, such as Strings, you must use the equals() method

1. [10 pts] Should a language require the declaration of variables? Languages such as Lisp and Python allow variables names to be used without declarations, while C, Java, and Ada require all variables to be declared. Discuss the requirement that variables should be declared from the point of view of readability, writability, efficiency, and security.

I believe that variables should be declared, mainly for clarity and **re-readability** sake.

**Efficiency:**

**Readability:** It makes much more sense to a programmer who did not write the program when the variable are declared. The more clearly defined a program is, the better and faster a programmer can understand its workflow/purpose (to a degree, avoiding over explanation).

**Writability:** Declaring variables may increase the overall time it takes to code, that time should be negligible, provided that the programming language’s variable declaration syntax is created with a programmer’s process in mind. Variable declaration also makes a program more concise by leaving less information for others to guess or be unsure about.

**Security:**

Variable declaration greatly contributes to the security of the program/programming language. By declaring the variable, you can directly specify the type. This will result in less type errors later on when using the variable.

1. [9 pts] List the two types of programming language abstractions and the three levels of programming language abstractions

The two types are: **Data Abstractions** (numbers, strings, etc.) and **Control Abstractions** (loops/conditionals)

The three levels are:

* Basic Abstraction
  + Variables, data types, math operations
* Structured Abstraction
  + Data structures
* Unit Abstractions
  + a standalone collection of procedures providing logically related services to other parts of a program