Nicholas Alexander
May 4, 2015
CS 300
Programming Languages

High Stakes 2 Response

1. For each Prolog, SAS, and C give an ideal application and a non-ideal application.

Prolog, being a functional language, is very useful for applications in the artificial Intelligence and data mining areas. It is easily demonstrated that Prolog is capable to forming complex relationships with very few base rules – a highly desirable trait in AI and machine learning. Machine learning wants to use base rules and generate new rules from them. However, because functional languages like Prolog are inherently stateless, the language would not be well-suited for interactive applications such as a word processor or a web browser.

SAS is a statistical analysis language meaning it has very niche uses. Most obviously, a statistician for a business would find SAS useful for analyzing demographics, sales, etc. However, SAS cannot be used for anything other than statistical modelling, so the language is very limited. The limitations, though, allow the statistical modelling capabilities to be extremely polished.

C is the one of the most popular and widely used programming languages in the world. Because it is imperative and has a large community of library developers, there are very few problems that C cannot be used to solve. The limitation to C is

not so much in its applications but rather in the portability of those applications. C programs must be compiled on the current machine to insure portability meaning that an application in C cannot be perfectly cross-platform. As stated before, C is a very powerful, optimized, and fast language so it is a great choice for any type of consumer application like a word processor, satellite software, or mainframe computing applications.

2. **Compare and contrast C and Java. Note the similarities and differences in syntax, when to use one language over the other, and dangerous pitfalls.**

The first thing even a layman would notice when comparing C and Java source code is the overall format. All Java files must contain a class whereas C programs at the least must contain a main function. This is because Java is an object oriented programming language which means all actions are done through classes and objects made from those classes. C, on the other hand, is a simple imperative language that relies on functions to do work – note that C is ***NOT*** a functional programming language. Interestingly enough, if you look inside a Java program you will find a method called main, just like the main function in C, where the program begins execution. Many languages use main as the starting execution point and this is likely due to the overwhelming popularity and adoption of C.

Much of the syntax between C and Java is very similar and thus Java is routinely called a “C-style” language. For example, in both languages all statements end with a semicolon; all code blocks are surrounded with curly

braces; arrays support the square-bracket access method; functions/method signatures are the same format; the list goes on and on. Just to reiterate, the major syntax difference between the two languages is that all Java files are classes.

When trying to decide whether to use C or Java, there are a handful of major considerations to make. First and foremost, one must consider if the object-oriented (OO) paradigm is better suited for the project rather than a straight imperative approach. Often times, OO principles allow a project to remain clean, modular, and organized. The object-oriented decision is very high-level and does not affect the overall outcome since both paradigms will achieve the same result. More importantly, one must consider the security risks that C exposes an application to whereas Java does not. Typical examples of these risks are buffer overflows, accidental or intentional memory tampering via pointers and unchecked array bounds, and error/exception handling. Each of the following will be explained in depth in the following paragraph.

A buffer overflow occurs when data is being written to memory but the allocated memory is not sufficient and more memory is used. For example, imagine that 20 bytes are allocated for a string containing a user's first name. If the user enters a garbage value that is 25 bytes, the original chunk of memory is not large enough so the program will wildly write over an additional 5 bytes of adjacent memory. This can be extremely problematic because those 5 bytes could have been *anything* including other program data or even operating

system data. C is extremely vulnerable to buffer overflows because there is absolutely no bounds checking. In fact, a buffer overflow, codenamed Heartbleed, was recently discovered in one of the most widely used open-source libraries, OpenSSL. Java, though, is much less susceptible to overflow since there is strict bounds checking and an exception will be thrown (more on this later).

Similar to buffer overflow, when using C, memory can also be tampered with using pointers and array accesses. A pointer is a memory reference itself meaning memory can be directly accessed with no safety checks whatsoever. Java does not give the programmer access to pointers so this is a non-issue. Array accesses in C also allow the programmer to overstep the allocated memory bounds. If an array is 10 elements long, C will gladly allow the programmer to write to the 11^{th} space and willingly overwrite whatever data is currently at that memory location. Java, however, will through an exception just like with a regular buffer overflow.

At this point, Java exceptions have been mentioned twice. In most object oriented languages, when an error occurs the program will raise an exception stating that something went wrong. Some examples of exceptions in Java include ArrayIndexOutOfBoundsException, when the program attempts to overstep the bounds of an array; NullPointerException, when the program tries to perform an operation on a null value; and BufferOverflowException, when a buffer overflow is detected. An exception allows a programmer to appropriately act based on a specific error and attempt to recover as opposed to simply crashing. Exception

handling is great because the programmer can never know what types of input a user may provide – either through accident, possibly penetration attempts, or simply 'to see what happens.' C does not have any type of error or exception handling built in. Instead, the programmer is expected to prevent any and all errors from ever occurring through type checking, length checking, and other explicit safety checks. That is an incredibly daunting, and possibly impossible, task.

The final major point to make between C and Java is the performance. C is extremely optimized and performs brilliantly when executed. The C compilers have been maintained by some of the greatest software developers of all time. C is lightning fast and is, quite possibly, unmatched in performance and efficiency. It is so efficient because it does not waste time doing senseless safety checks (deferring to the programmer) and allows direct, speedy memory manipulation. Java, though, is much slower because it runs inside the Java Virtual Machine (JVM). While a discussion on the inner-workings of the JVM is beyond the scope of this paper, suffice it to say that Java is ran in a type of virtual environment which introduces a sizeable performance overhead. Keep in mind, though, that this discussion is on the order of nano to milli seconds of difference between the two languages' execution times of small-scale programs.

While this is far from an exhaustive list of the differences between C and Java, it summarizes some of the most crucial factors to consider when choosing between the two languages. One must further understand that there is no right

or wrong programming language for any given task; instead, there are simply "better" or "worse" languages. Anything that can be done in Java can be done in one way or another in C and vice-versa.

3. **A researcher has a database filled with patient records each of which include demographic information as well as medical measurements. The researcher desires to find people with undiagnosed diseases. Which language would be best suited for the task: Prolog, C, or SAS? Explain.**

   One approach to answering this question is through elimination. Immediately, it is known that SAS is mostly a statistical modelling language and is probably not the best suited for this type of data mining operation. This is not to say that SAS could not meet the researcher's needs, since it is (arguably) a Turing complete language. C, of course being a powerful Turing complete language, is capable of solving nearly any problem. The problem, however, is that C is a procedurally imperative language which is not ideal for data mining tasks. Through elimination we are left with just Prolog. Prolog is a very powerful functional programming language which is great for data mining tasks because it can infer new relationships based on a set of smaller, more-obvious relationships. The research has records containing medical data – the smaller, obvious relationships – and has knowledge of how these symptoms correlate into diseases – the new relationships.

4. **Another researcher is interested in the effect that the gender demographic has on diseases. Given the data set, explain the language and approach to generating a meaningful report.**

   Anyone that is experienced with the language will immediately see that SAS is a prefect choice for this task. The researcher wants to perform a statistical analysis on a data set and produce a useful, formatted report. This is a perfect task for SAS and procedures such as 'freq' and 'means.' Not only do the SAS procedures do all the number-crunching in a black-box, but the procedures also produce an easy-to-read representation of the data.

5. **Common language choices for Computer Science I at many universities are Java, C++, C#, and Python. Which language would you choose and why?**

   The first language that a programmer learn often sticks with them for the remainder of their career in one way or another. If a student is taught object oriented (OO) languages from the beginning of their studies, that student will likely stick with other object oriented languages throughout their career as opposed to shifting to another non-OO language. When picking the first language to teach, there are a number of matters to take into consideration. First and foremost is the simplicity of the language. One does not want to deter students from learning the principles of programming by fighting with convoluted syntax. Next is the concern of licensing since not all languages are freely available. Finally, there is the career implications of knowing one language over another since not all languages are popular in industry. Each of these concepts will be discussed further.

Any newbie or veteran programmer will say that a language whose syntax is easy to read and understand is far superior to those whose syntax is clunky and difficult to decipher. For beginners, a language whose syntax closely resembles the English language is always best. Furthermore, languages that do not use many symbols - like curly braces, brackets, semicolons, etc. – are often easier to understand. For this reason, Python is a great language for beginners. Python has a very minimalistic type syntax but it also leaves out concepts such as static typing that is prevalent in most other languages. C# and Java have very similar and very verbose syntaxes. One line in Python would often require upwards of 10 lines in Java or C# due to class definitions and such. C++ is notorious for using a number of symbols, such as the stream operators '<<' and '>>' which can be daunting to new users. In this area, I think Java or C# would win. Python is a close second – the lack of explicit typing is an extreme drawback.

Because universities, at their core, are businesses and want to make money, they are not always apt to spend thousands of dollars to get an enterprise license of a compiler or software system. For example, before the days of GCC and Clang, C/C++ compilers were not free and were quite expensive. Other languages, such as SAS, are also licensed to enterprises for large sums of money. Because Java, C#, C++, and Python are all free they are often prime choices.

Finally, there is the concern of whether or not a language will be potentially used in the industry. If a student learns a language that no one in the world uses, that knowledge is rather useless. Instead, universities strive to teach students the most popular languages – Java, C#, C++, and Python. Java is extremely popular since it is cross platform and strongly endorsed by big businesses such as IBM; C# is also very popular since it aligns with Microsoft's essential .NET framework; C++ is the natural successor of C and is used in many high performance computing areas; and Python is a used in scientific computing as well as web development. Each of the 4 languages have a strong following; in fact, the statistics show that they are all within the top 10 most widely used languages on GitHub. The only languages that match Java, C#, C++, and Python are the hardcore web languages such as JavaScript and PHP. The problem with the web languages, though, is that they are often scripting languages which are somewhat less powerful than application languages. It is safer to learn an application language and port that knowledge to a scripting language.

Overall, if I had to choose the next language to be taught at Marshall University, I would pick either Java or C#. Java is so widely used that it will likely not disappear for years to come. C# has a strong background with the .NET framework that makes it extremely viable in industry. Both of the languages are object oriented which is the norm in the industry today. Both languages are also free to use. The best bet, though, would be to stick with Java since C#

development is largely Windows-based. Because Java is cross-platform, all Windows, Mac, and Linux users will have no problems with it.

6. Assign yourself a grade for each of the problems 1 through 5.

1) A because each language is deeply analyzed and a strong application area and a weak application area are highlighted.
2) A because both of the languages are deeply compared based on three largely important concepts.
3) A because the argument clearly expresses why one language is better than the other two for the given task.
4) A also because the argument clearly expresses why one language is better than the other two for the newly given task.
5) A because each language's pros and cons are discussed as well as some external factors.