**Design Document**

For testing Ackerman Function benchmarks read the README.txt

**Benefits / Limitations of C language:**

The C language is the “current” language being taught by the University and has the most documentation (easy to find / access), making this the easiest and highest benefit over all the other languages. If I was brought up using Ada, Fortran, or Cobol they would probably be the language I would choose as the best language. Testing with the Ackermann function I used a high input such as Ackermann(m,n) : (3,12) and my result shocked me where C performed the quickest on the Linux machine with 6 seconds, also the iterative version calculated it in 12.7 seconds which is still the fastest with my calculations.

**Some benefits of using C:** It is easy to find documentation, huge amount of updated documentation, everyone I know understands C easily and can help me easily, I understand C the most. Finding a job would be easiest in C as it is the modern day language to use and it would be harder to find a job in the other legacy languages.

**Limitations of using C:** I am not aware of many limitations of using C, it is a strong language that is very close to assembly level ( just a step above it) making it a very optimized and huge performance boost compared to other languages. Some limitations of using C are that it does not handle memory very well and it’s dependent on the programmer to clean everything up and it’s still a very difficult task to do. There are a lot of security issues with C and that’s why some of the other Legacy languages below are still being used over C.

**Benefits / Limitations of Ada language:** Ada is a legacy language, and I had to learn it for this class. It was a very overwhelming language that I ended up grasping it very well after a few days. I tested Ada with Ackermann function in the recursive and non-recursive version to all the other languages. When I used the Ackermann function (m, n) : (3 , 12) my result came out as 10 seconds using recursion on the linux machines, which was the second fastest result just after C. For the non-recursion test it came out as 21 seconds making it the 2nd fastest language using non-recursion compared to C, Fortran and Python.

**Benefits of using Ada:** Programs written in Ada have fewer bugs and are delivered faster than those written in C (modern day language to compare to). Ada is generally used for embedded systems, because it targets real time embedded systems, it’s faster than C with embedded systems, and its free. Programs are designed to “think in real time” with Ada projects using any embedded systems, which are used for life critical companies ( hospitals, military).

**Limitations of using Ada:** There are limited school’s/teacher’s that teach Ada knowledge and experience. It is a legacy language which is quickly getting older and older, with documentation that does not compare to modern day languages like C. It is harder to find “the right” documentation/examples for a specific task compared to C. It is much harder for new developers to get a hang of Ada and most people get turned off quickly (myself included), it feels very overwhelming quickly compared to modern day languages.

**Benefits / Limitations of Fortran language:**

Fortran is a legacy language, and I had to learn it for this class. It felt very easy to grasp and similar to Lua which made it easy to learn. I tested Fortran with Ackermann function in the recursive and non-recursive version to all the other languages. When I used the Ackermann function (m, n) : (3 , 12) my result came out as 14.69 seconds using recursion on the Linux machines, which was the third fastest result just after Ada. For the non-recursion test with (3, 12) came out as 26.23 seconds, and this was the slowest time! This shocked me because Fortran is suppose to be the “mathematics” based language and it did not perform as well compared to C and Ada.

**Benefits of using Fortran:** Fortran is designed to make numerical computation easy, robust and well-defined, is it more efficient in mathematics compared to the other languages. It is easier to learn than C because it much simpler version of the code. Tons of existing Fortran code which is publicly available. The compiler in Fortran generally output’s much better messages compared to the other languages. If I started with the modern day languages (c, Java, C++ , etc), programming in Fortran was easy to grasp and learn very quickly.

**Limitations of using Fortran:** In the real world it is very hard to find a job with using Fortran, it would only be useful for research in an University environment. It is a “legacy” language which is not used very much anymore. Compared to C it was very hard to find the documentation that I “was looking for” and made starting with Fortran very hard. There is not enough documentation out there compared to the modern day languages.

**Benefits / Limitations of Python language (switched from COBOL – allowed from Prof):**

Python is a modern day language, which I learned while I was at school for another project. It is a general purpose programming language that is very easy to learn, but the cost is being very slow compared to the other languages mentioned above. I tested Python with Ackermann function in the recursive version to all the other languages. When I used the Ackermann function (m, n) : (3 , 11 \*using 12 caused segmentation fault\*) my result came out as 154.3 seconds using recursion on the Linux machines, which was by far the slowest language! Iterative version was not needed because the professor said we could swap python for Cobol and only one version was needed.

**Benefits of using Python:** Python is a general purpose programming language, it can be used for almost anything and not an expert at anything. Python is also very readable and easy to learn for any new programmer. It is a very flexible language and does a good job for any new programmer to start with Python as their first language. Python is a good tool for any start-up company as it is very productive and easy to prototype new ideas. It is still a modern day language with enormous amount of documentation and easy to find examples for anyone.

**Limitations of using Python:** Python is not a good tool for CPU extensive applications as it is not as fast compared to the other languages out there. Not as many Python jobs out there compared to C. It is very slow compared to the other languages that do it much better, for example when testing the Ackermann function Python was unable to even remotely come close to C or any other language.

**Q & A:**

• **Which program had the best usability(ex. was easiest to program, had the most appropriate structures, is easy to maintain)?**

The C programs were by far the easiest to program, and that’s only because that’s what University has taught me to learn over the past few years. It was by far the easiest to maintain and the most appropriate structure that made sense to me. Since I knew C for a while now it was easy to complete the task for this language and after completing the C part the assignment started to get difficult.

Right below C in the easiest to write was Fortran, the language was more natural with handling Ackermann function and numbers and I was able to quickly complete the goal. This language felt very similar to the C structure and felt natural writing Ackermann function in Fortran. It was very easy to maintain and I felt like it had appropriate structures that make sense.

Under Fortran was Ada, because it is still very similar to C. I just had to understand how to properly call functions / algorithms and once I was able to figure that out writing Ackermann function in Ada was not that difficult. It was not as easy to maintain and it’s structure was a little more confusing to me compared to Fortran/C.

The most difficult program to write and maintain by far was Cobol. I had a very difficult time writing the program, and I still don’t understand Cobol very well. Since I don’t understand Cobol very well it made writing Ackermann function very difficult!

• **Was there any difference in efficiency for these programs?**

Yes there is a big difference in efficiency compared to all the programs together. I decided to use Ackermann function(m,n): 3,12 as my input for all my tests to all the programs, the results I got were an eye opener for me. Recursion and iterative versions of the same language had huge differences in the time it took to complete!

The recursion version of all the languages, C was the quickest to complete at 6 seconds, the next in line was Ada at 10 seconds, the third one was Fortran at 14.69 seconds, and the last one was Python at 154.3 seconds (3,11 as the other input caused a segmentation fault! ). I thought Fortran would be the quickest as the language is suppose to handle numbers better and the results showed Fortran was one of the slowest languages!

The iterative version of all the languages had incredibly different results compared to recursion. For instance the C program in recursion took 6 seconds, but using the same numbers in the iterative (stacks) it took 12.7 seconds to complete! It took double the time to complete when I changed from recursion to iterative! Another example of comparing C iterative version to Fortran using (3,12) the C result was 12.7 while the Fortran result took 26.23, almost double the time. This example shows a huge difference in efficiency to using these languages! From the fastest to slowest in times using ackermann function (m,n) : (3,12), it was C at 12.7 seconds being the fastest, next was Ada with 21 seconds, and last place was Fortran with 26.23 seconds. I used recursion Python so there was not a 4th test for iterative.

It appears when using recursion for longer/higher calculations that the recursive formula is the quickest. But when using smaller calculations for the Ackermann Function the stack method would be quicker.