CSCI 3415 – Programming Languages
Dr. Williams
Programming Assignment 1 – Julia
Due Oct. 1, 2018 by midnight

In this programming assignment, you will implement the example program previously discussed, which inputs a sequence of integers, computes the average of the numbers, and prints the number of numbers greater than the average, in Julia.

Julia

Julia is a high-level general-purpose dynamic programming language that was originally designed to address the needs of high-performance numerical analysis and computational science, without the typical need of separate compilation to be fast, also usable for client and server web use, low-level systems programming or as a specification language.[1]

Distinctive aspects of Julia's design include a type system with parametric polymorphism and types in a fully dynamic programming language and multiple dispatch as its core programming paradigm. It allows concurrent, parallel and distributed computing, and direct calling of C and Fortran libraries without glue code.

Julia is garbage-collected, uses eager evaluation and includes efficient libraries for floating-point calculations, linear algebra, random number generation, and regular expression matching. Many libraries are available, and some of them (e.g. for fast Fourier transforms) were previously bundled with Julia.

All told, Julia is:[2]

- **Fast**: Julia was designed from the beginning for high performance. Julia programs compile to efficient native code for multiple platforms via LLVM.
- **General**: It uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns. The standard library provides asynchronous I/O, process control, logging, profiling, a package manager, and more.
- **Dynamic**: Julia is dynamically-typed, feels like a scripting language, and has good support for interactive use.
- **Technical**: It excels at numerical computing with a syntax that is great for math, many supported numeric data types, and parallelism out of the box. Julia's multiple dispatch is a natural fit for defining number and array-like data types.
- **Optionally typed**: Julia has a rich language of descriptive data types, and type declarations can be used to clarify and solidify programs.
- **Composable**: Julia's packages naturally work well together. Matrices of unit quantities, or data table columns of currencies and colors, just work — and with good performance.

[1] Wikipedia contributors. "Julia (programming language)." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 16 Sep. 2018. Web. 17 Sep. 2018.
[2] "Julia 1.0", The Julia Blog, 08 Aug 2018

Specifics

Download and install the Julia 1.0 system. Julia is available on Windows, Apple OS X, and Linux operating systems. You may use whichever of these you choose.

Julia code at this level can easily be created and edited using any text editing system. However, there are various IDEs available if you would like to use one. I personally use Juno (http://junolab.org/) as a Julia IDE – it is available on the platforms above, but I have not used it except on Windows.

You are to write a Julia program equivalent to the example program discussed in class and provided in various languages on canvas. [For example, there are two different C++ implementation given.]

Here is one of the C++ programs:

```
/* C++ Example Program
 Input:  An integer, listlen, where listlen is less than
         100, followed by listlen integer values
 Output: The number of input values that are greater than
         the average of all input values */
#include <iostream>
#include <cstdlib>
using namespace std;
int main() {
      int intlist[99], listlen, sum = 0, average, result = 0;
      cin >> listlen;
      if ((listlen < 1) || (listlen > 99)) {
            cout << "Error - input list length is not legal" << endl;
            return EXIT_FAILURE;
      }
/* Read input into an array and compute the sum */
      for (int counter = 0; counter < listlen; ++counter) {
            cin >> intlist[counter];
            sum += intlist[counter];
      }
/* Compute the average */
      average = sum / listlen;
/* Count the input values that are > average */
      for (int counter = 0; counter < listlen; ++counter)
            if (intlist[counter] > average) ++result;
/* Print result */
      cout << "Number of values > average is: " << result << endl;
}
```

Letter Grade	Description
A	90 and above
B	80 – 89
C	70 – 79
D	60 – 69
F	Below 60

Report

You must include a short write up about your program – that is all you will turn in (as a PDF file). It should include a description of the program – essentially what is contained in this document, your design, the implementation details (including the code), sample runs, and conclusions. It should also cite any sources you used in developing your program.