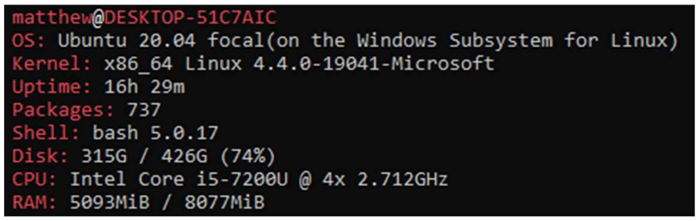
Assignment 1 – u20734621

Matthew Gotte

System Information



```
matthew@DESKTOP-51C7AIC
OS: Ubuntu 20.04 focal(on the Windows Subsystem for Linux)
Kernel: x86_64 Linux 4.4.0-19041-Microsoft
Uptime: 16h 29m
Packages: 737
Shell: bash 5.0.17
Disk: 315G / 426G (74%)
CPU: Intel Core i5-7200U @ 4x 2.712GHz
RAM: 5093MiB / 8077MiB
```

All tests were executed and timed by tester.sh on the Windows Subsystem for Linux.

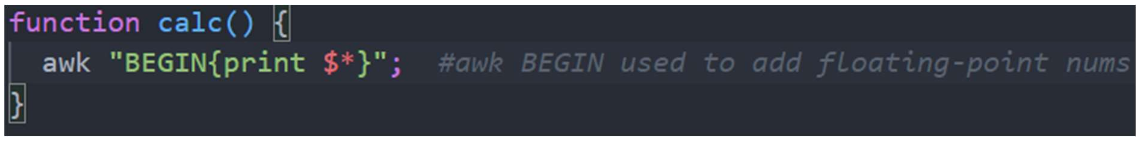
Script/Program to execute the experiment:

Explanation of approach:

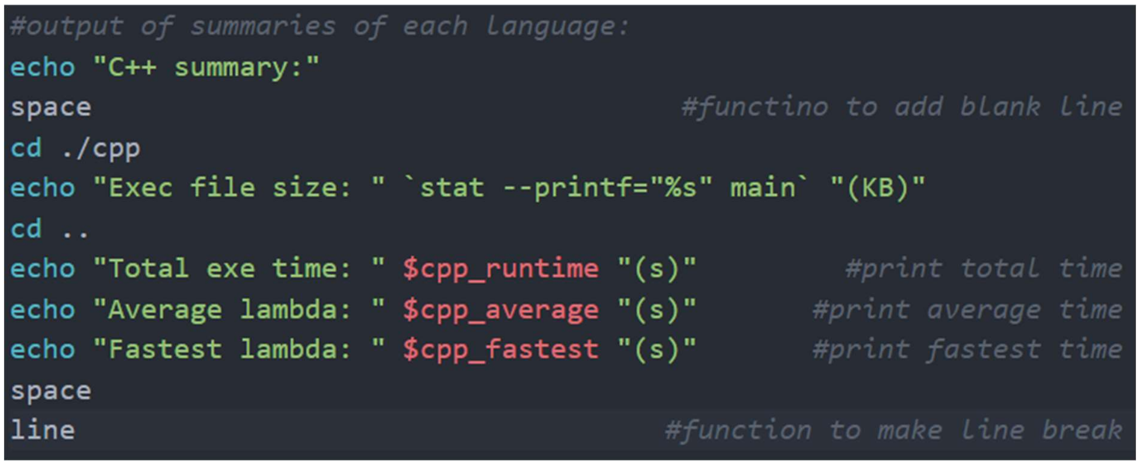
The script begins with calling the functions to begin the tests on each language respectively. The test comprises of generating a λ, (with λ = single execution x 500). Each lambda was recorded and added to a running total to maintain the total execution time of the language, as well as compared to a variable to maintain which λ (500 executions) executed the fastest. The script repeats this for each language and then generates the following:

1. Executable file size in KB.
2. Total execution time for all λ in seconds. This total is purely the execution time of the executable file and does not factor any overhead code such as comparison in each iteration to maintain the lowest λ, this makes it very accurate in terms of the total execution time of the language.
3. Average λ for that language in seconds. This value is given as (Average λ) = (Total λ) / 50, due to 50 λ being calculated
4. The fastest λ that executed in seconds. This is managed by assuming first λ being fastest then a comparison with each λ generated after and maintaining a value of the lowest λ that executed

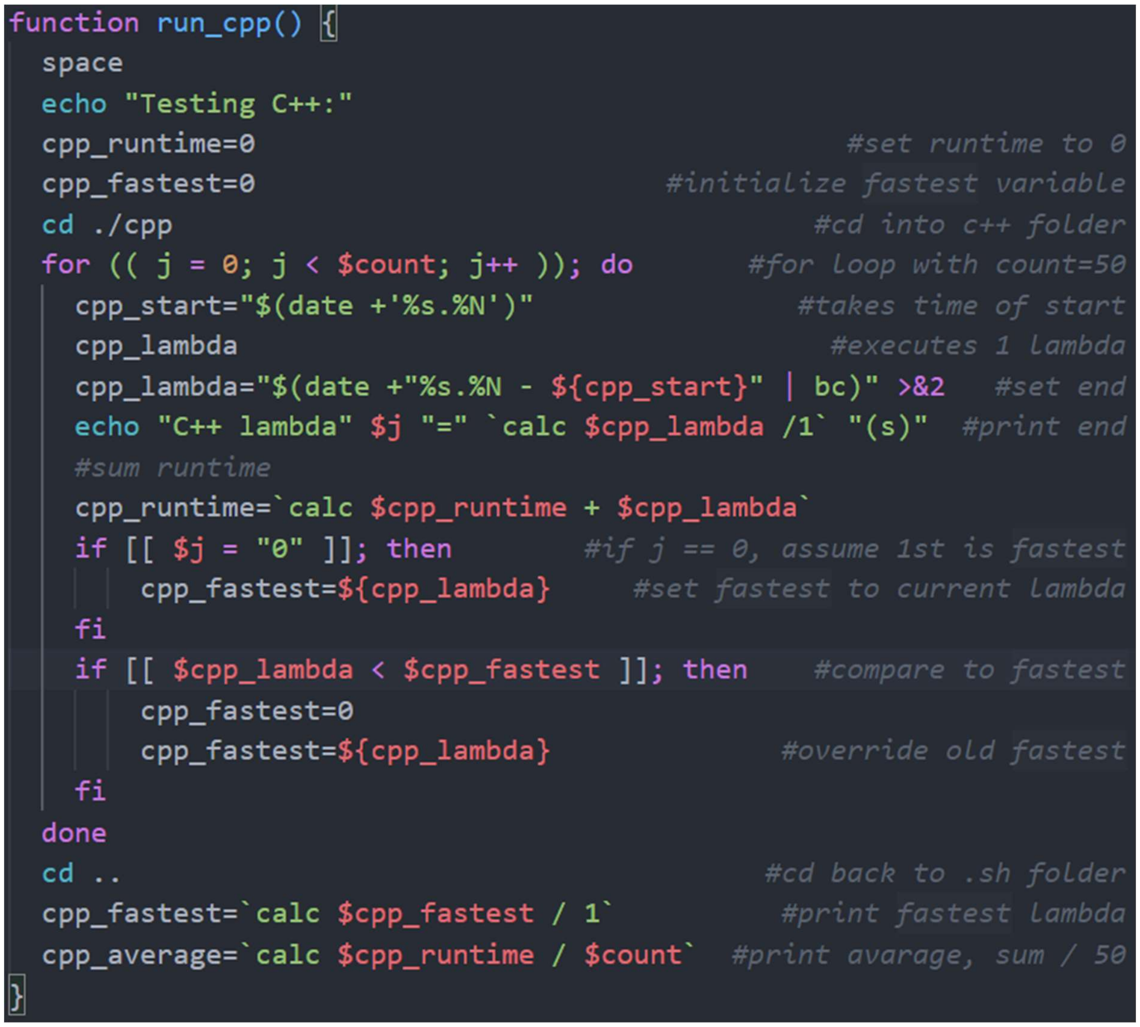
The Script (refer to comments in script for explanations):



```
function calc() {
  awk "BEGIN{print $*}";  #awk BEGIN used to add floating-point nums
}
```

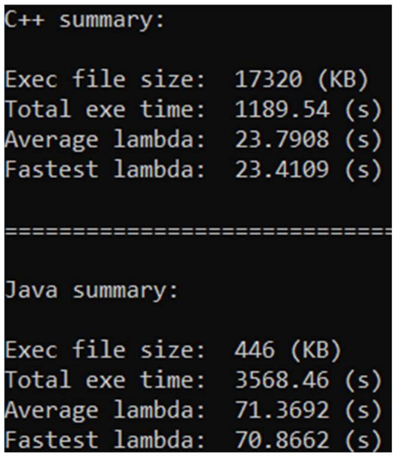


```
#output of summaries of each language:
echo "C++ summary:"
space                                      #functino to add blank line
cd ./cpp
echo "Exec file size: " `stat --printf="%s" main` "(KB)"
cd ..
echo "Total exe time: " $cpp_runtime "(s)"         #print total time
echo "Average lambda: " $cpp_average "(s)"       #print average time
echo "Fastest lambda: " $cpp_fastest "(s)"       #print fastest time
space
line                                        #function to make line break
```

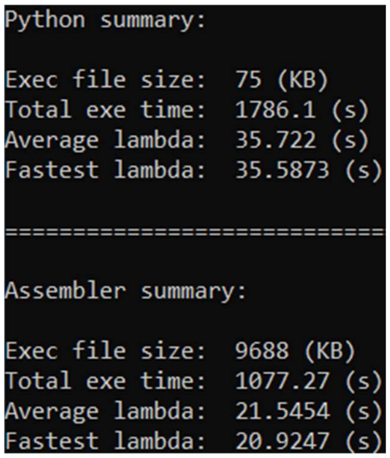


```
function run_cpp() {
  space
  echo "Testing C++:"
  cpp_runtime=0                                              #set runtime to 0
  cpp_fastest=0                                      #initialize fastest variable
  cd ./cpp                                                #cd into c++ folder
  for (( j = 0; j < $count; j++ )); do                  #for loop with count=50
    cpp_start="$(date +'%s.%N')"                           #takes time of start
    cpp_lambda                                               #executes 1 lambda
    cpp_lambda="$(date +"%s.%N - ${cpp_start}" | bc)" >&2           #set end
    echo "C++ lambda" $j "=" `calc $cpp_lambda /1` "(s)"        #print end
    #sum runtime
    cpp_runtime=`calc $cpp_runtime + $cpp_lambda`
    if [[ $j = "0" ]]; then            #if j == 0, assume 1st is fastest
        cpp_fastest=${cpp_lambda}           #set fastest to current lambda
    fi
    if [[ $cpp_lambda < $cpp_fastest ]]; then          #compare to fastest
        cpp_fastest=0
        cpp_fastest=${cpp_lambda}                      #override old fastest
    fi
  done
  cd ..                                                   #cd back to .sh folder
  cpp_fastest=`calc $cpp_fastest / 1`                    #print fastest lambda
  cpp_average=`calc $cpp_runtime / $count`      #print avarage, sum / 50
}
```

<u>Experiment Results:</u>



```
C++ summary:

Exec file size:  17320 (KB)
Total exe time:  1189.54 (s)
Average lambda:  23.7908 (s)
Fastest lambda:  23.4109 (s)


=============================

Java summary:

Exec file size:  446 (KB)
Total exe time:  3568.46 (s)
Average lambda:  71.3692 (s)
Fastest lambda:  70.8662 (s)
```



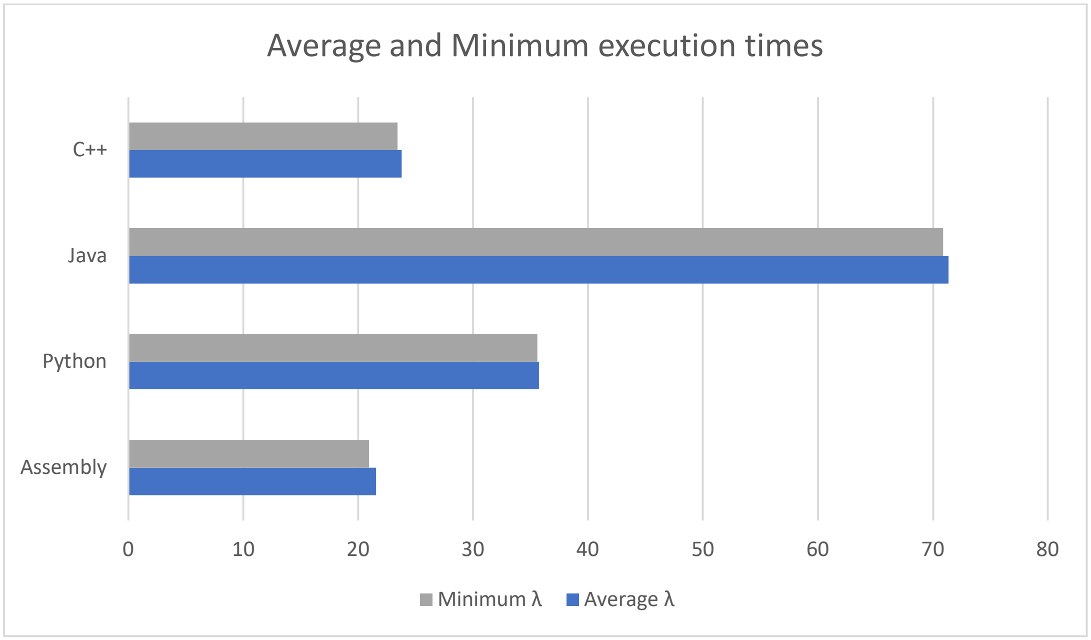
```
Python summary:

Exec file size:  75 (KB)
Total exe time:  1786.1 (s)
Average lambda:  35.722 (s)
Fastest lambda:  35.5873 (s)


=============================

Assembler summary:

Exec file size:  9688 (KB)
Total exe time:  1077.27 (s)
Average lambda:  21.5454 (s)
Fastest lambda:  20.9247 (s)
```

Table of results:

	C++	Java	Python	Assembly
Total time [50 λ] (s)	1189.54	3568.46	1786.1	1077.27
Average λ	23.7908	71.3692	35.722	21.5454
Minimum λ	23.4109	70.8662	35.5873	20.9247
Average per 25000 executions	0.0475816	0.1419384	0.071444	0.0430908
File size (KB)	17320	446	75	9688

Note: λ = 500 executions of a language Average per execution = Total time ÷ (50 x 500)

Graph of results:



Conclusion and Observations:

The slower languages (comparatively speaking) were the languages that compile into bytecode and are then interpreted. This presents in the results with Java and Python being the two slowest languages of the four.

The faster languages are the ones that are compiled languages, languages in this category are compiled into machine code by the compiler. This makes them faster than those converted to bytecode then interpreted. This presents in the results of C++ being faster than Java and Python.

The fastest language (according to the results produced in this experiment) is Assembly, this is due to assembly being purely assembled opposed to being compiled. Assembly is the lowest level language out of the four languages that were tested and thus it has shown that a purely assembled language is faster. With the results of this experiment, it can be said that in terms of increasing execution time, the order is, Assembly, C++, Python then Java.