**Spike:** Task 5

**Title:** IDE Familiarity

**Author:** Sam Huffer, 101633177

**Goals / deliverables:**

* Demonstrate (through this spike report) the following performance and measurement concepts:
  1. Measurement of single and multiple functions’ execution times.
  2. Linear and exponential ramp-up testing of function execution time, comparing to ramp-down testing.
  3. How repeatability will vary depending on test conditions.
  4. Comparison of functions providing the same functionality.
  5. Whether there’s any difference in execution time between debug and release settings in Visual Studio.
  6. Whether turning down / off compiler optimisation makes a difference.
* Record and chart the data collected for each section.

**Technologies, Tools, and Resources used:**

* Visual Studio 2017
* Microsoft Word
* Microsoft Excel

**Tasks undertaken:**

1. *Single Tests: Demonstrate how to measure both single and multiple function execution time.*
   * .
2. *Ramp-up Test: Execute and show (numbers/make a chart) both linear and exponential ramp-up testing of function execution time. Is there a difference to ramp-down tests?*
   * Had a look at the code for both of the test methods, ran the code, and recorded the results in an Excel spreadsheet.
3. *Repeatability: Show (with numbers/chart) how repetability will vary depending on test conditions.*
   * *.*
4. *Function Comparison: There are two "char in string” counting functions provided (code sample 1). Clearly show the difference in performance (if any), and show if the speed difference is linear to string size. (Note, you will probably want to create random strings of the various size to test with.)*
   * Looked at the code for ramp-up testing, taking how it recorded the time it took to execute a chunk of code, and implemented that around the count\_char…() methods. Ran the code several times using strings of different lengths, and recorded the results.
5. *IDE Settings: Show what, if any, is the difference in execution time between debug settings and release settings. (Remember to have a task that runs for long enough that it matters.)*
   * .
6. *Compiler Settings: Turn down/off compiler optimisation and demonstrate a difference.*
   * .

**What we found out:**

* Shortcut keys (where available) for the aforementioned tasks.
* Refreshed my understanding of how to work with breakpoints.
* How to view the values of variables at a particular line, within a method, and chosen variables across the program.