Practical Lab Session: Week 7

For this practical lab session, please first ensure you have viewed the **Week 7** video sessions **7.1**, **7.2** and **7.3**. Within these sessions, you should attempt the development exercises presented as **Challenges** during this lab session. Please ensure you complete the set of challenges prior to the next lab session and upload screenshots of your results to the Progress Management section on Blackboard as directed.

Session 7.2 Challenge: Shell Sort

- Measure the performance of the Shell Sort technique
  - Revisit the previous challenge and measure the execution time to sort 1000 arrays of 100, 200, 400, 800, 1600, 3200 and 6400 elements.
  - Report the results as a series of 2 values per line – array size and sort time.)
- The selection of gap sequences for Shell Sort is a topic of hot debate and many alternatives have been proposed.
  - Compare some of the most popular gap sequences by measuring the time to sort 1000 arrays of 10000 elements, using the following sequences…

    n/2, n/4, n/8, …, 1 (as in our algorithm)
    n/3, n/6, n/9, … 1 (divide by 3 rather than by 2 at each stage)
    n/4, n/16, n/64, … 1 (divide by 4 at each stage)
    3785, 1695, 749, 326, 138, 57, 23, 9, 4, 1 (mean of prime numbers sequence)

  - Show the execution time for each gap sequence

Note: an example of the output from the program is provided in the video session.

Session 7.3 Challenge: Merge Sort

- Measure the performance of the recursive Merge Sort technique
  - Revisit the previous challenge and measure the execution time to sort 1000 arrays of 100, 200, 400, 800, 1600, 3200 and 6400 elements.
  - Report the results as a series of 2 values per line – array size and sort time.)
- The Implementation note earlier suggested that the temporary array be passed to the `merge()` method as a parameter. Measure how effective this tip is in practice, by timing 1000 sorts of arrays of 5,000 and 10,000 integers

  i) With the `temp` array passed as a parameter (as in our implementation)
  ii) With the `temp` array generated anew each time the `merge()` method is called. Remember to remove `temp` from all parameter lists for this test.

Note: an example of the output from the program is provided in the video session.