The runtime for the computation time complexity for the DFT/IDFT pairs is $O\ of\ f(n^2)$. This is the case because the highest time complexity part of our algorithm is the 'for loop' for the Fourier Transform. Since the for loop is nested, the duration of one run is n*n, where n is the size of the signal samples array. For a DFT/IDFT pair, the algorithmic expression would be $2 \times n^2$. However, when looking at the upper bound as n tends to infinity, the constant 2 becomes irrelevant.

For the Second take home exercise, we noticed that the segmented DFT is almost four times faster than the non-segmented DFT. The samples size is 1024 and they were segmented into four for the first DFT. The run times are as follows: 241ms for 1024-point DFT and 70ms for four 256-point DFTs. The second method is significantly faster because the runtime scales exponentially with the size of the input array. By doing multiple iterations of smaller segment of samples, we can reduce the overall runtime significantly. For example: first method will perform 1028^2 = 1,048,576 operations whereas the second method will perform 4*256^2 = 262,144 operations.

For the third take home exercise, we implemented the block processing in C++ similar to the methodology used in Lab 1 where we used Python. Some of the challenges we faced were related to manipulating vectors in C++ such as concatenating them and providing a section of a vector as an argument into a function as a new vector.