Exercise 1:

To implement this problem we generated 1D arrays and merged four distinct calculations to make a separate 1D array. Within our define.h file we modified the array size to 1024. After this we needed to generate 2 separate arrays using gen\_array\_0 and gen\_array\_1. The generated arrays had to meet certain ranges and that was done using a simple modulus algorithm. To implement the 4 different quadrants of the new array C we renamed the existing 4 functions that were initially used to find the min and max. Within each quadrant function, the implementation would pend on specific mailboxes to start. For example the top\_left quadrant would pend on MboxTopLeftStart[0] and MboxTopLeftStart[1] which are posted once the generation of array A and array B is complete. Since we are working on a 1D array, to compute the values within each quadrant we implemented a linear algorithm depending on which value had to be multiplied together. Once all 4 quadrant multiplications were complete, we posted into the mailbox of each quadrant; MboxTopRightDone for top right quadrant. The final merge function would pend on the four done mailboxes to merge the four quadrants together. To generate array A and B it took 26053624 clock cycles, for the calculations to complete it took 68827058 clock cycles so total clock cycles before able to print array C is 94898433. These performances are accurate because generation of the two arrays is significantly smaller than computing the third array through multiplication. This is mostly because of the multiple nested for loops within each quadrant function.