**Implementation**

I decided to divide the exercises into three classes. The first one is the class for the matrix, which contains methods to get and set the matrix’s values to get the numbers of columns and rows. Then, two friend methods are printMatrix and multiply.

The second class is called SOE (=Sieve of Eratosthenes), which has methods to set limit and print the array of primes numbers.

Finally, there is the class Sortable which contains methods to create arrays, print the class, set and get the array, and get the size of the array and the ax value into it. The friend methods are the sorting algorithms.

**Considerations**

While implementing the sorting algorithms, it is possible to see a limitation in the bucket and count sort. As we can see, at the beginning of the method, both algorithms create a new array with a length of the max number in the array. It is a limitation because we cannot put in the sample array huge numbers. After all, if the number is significant, the system will create a segmentation fault.

Looking at the test results on all three algorithms, we can see that the bubble sort is the slowest, and the count and bucket sort spend much less time sorting the array. This is because the bubble sort has a complexity of O^2; on the other side, both count and bucket sort have a complexity of O. The limit for the bubble sort is the array size. The bigger it is, the more significant the time the algorithm will spend sorting the array.

I found no limitations in the matrix multiplication and the Sieve of Eratosthenes algorithm. The only problem, like the count and bucket sort, is the array's size. Also, the time that these two algorithms spend executing is relatively low.