For this assignment, you will create a series of functions that will operate on data contained inside an array:

```
array_double_raw(a, length)
array_double(a)
increment(a, i)
swap(a, i, j)
add(a, b)
subtract(a, b)
scalar_multiplication(c, a)
```

The first four functions will allow you to practice manipulating an existing array. The array_double_raw function takes a bare array and its length as parameters and doubles every element in the array. In contrast, the other functions should expect the length to be included as the first element in the array. This will allow your software to do runtime bounds checking. The increment function increments one element in the provided array. The swap function swaps the data at index i and j. The last three functions perform standard vector arithmetic operations and will expect you to create a new array to return your results in.

You should not need to modify the test suite. You only need to write the functions. Since QT SPIM expects all of your code to be in a single file, you can concatenate them together in a few ways. If you are on Windows, you can use the included batch file to do the work for you. Simply dragging your source file and dropping it on the batch file should be sufficient. If you are having trouble with the batch file, make sure that your file names match those below. You can also use a command line operation.

Windows: copy /Y “<Your Source File Name>”+”Test Suite.asm” Output.asm

Unix: cat “<Your Source File Name>” “Test Suite.asm” > Output.asm

Your program should include appropriate comments indicating what the code should be doing and what registers are being used for. After displaying the results, your program should exit cleanly. You should test your programs using the SPIM simulator to ensure their functionality before submitting them. You should only submit your functions. You will not receive credit if you submit the test suite in any form. You should also not include any driver or debug code in your submission.

Objectives:

1. To introduce arrays.
2. To understand array iteration.
3. To understand array indexing.
4. To review dynamic memory allocation.
5. To review parameter checking.