# Prelab 5: Arrays Continued

CSE/IT 113L

NMT Department of Computer Science and Engineering

---

Sometimes it is the people no one can imagine anything of who do the things no one can imagine.

— Alan Turing

If we do not share our stories and shine a light on inequities, things will not change.

— Ellen Pao

Most software today is very much like an Egyptian pyramid with millions of bricks piled on top of each other, with no structural integrity, but just done by brute force and thousands of slaves.

— Alan Kay

---

# Contents

# 1 Introduction

In this prelab you will write functions that operate on arrays.

# 2 Overview

In order to receive full credit for this lab, you must read the following material and follow the directions for coding with structures. Prelab 5 is a review of arrays.

# 3 Lab Specifications

The following subsections describe what you will write and submit for this lab. **Check that you have met all of the requirements for this lab before you submit it for grading, as the lab will be counted as late if you submit or resubmit it after the due date.**

## 3.1 Array Problems

**Exercise 1** (vector.c, vector.h, v.script, main.c).
From the tarball, define functions for problems 1 - 8 in `vector.c` and add prototypes for those functions to `vector.h`. Use `main.c` for your `main()` function.

Write a script called `v-output.script` that uses **the same arrays as what are given in the example.**

All values should be hardcoded and the functions should run in the order that they are given in this lab. **If values are given to you make sure to use them. Output should be identical to what is shown below.**

Function names are a design decision, however, function names should reflect what the function does.

Please write your functions `in order` with the first one to be ran at the top of vector.c and downward from there. Your function prototypes should also match this order.

1. Write a function that multiplies each element of an array by an integer, **5** ,and stores the new value in the same array. Please print data to the screen like shown below:

   *Before*

```
a1[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
```

*After array is multiplied by **5***

```
a1[] = {5, 10, 15, 20, 25, 30, 35, 40, 45, 15}
```

2. Write a function that adds an integer, 3, to each element of an array and stores the new value in the same array.

   *Before*

   ```
   a2[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   ```

   *After adding **3** to the array*

   ```
   a2[] = {4, 5, 6, 7, 8, 9, 10, 11, 12, 6}
   ```

3. Write a function that copies the contents of one array into a new array. Both arrays have to be the same size.

   *Before*

   ```
   a3[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   b3[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}
   ```

   *After copying the first array into the second*

   ```
   a3[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   b3[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   ```

4. Write a function that finds the sum of elements for each position within the two arrays and writes the results of the summations to a third array. Since functions can't return arrays, you have to pass in three arrays to your function. All arrays must be the same size.

   *Before*

   ```
   a4[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   b4[] = {10, 10, 10, 10, 10, 10, 10, 10, 10, 10}
   ```

   *After adding the arrays*

   ```
   a4[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
   b4[] = {10, 10, 10, 10, 10, 10, 10, 10, 10, 10}
   c4[] = {11, 12, 13, 14, 15, 16, 17, 18, 19, 13}
   ```

5. Write a function that finds the product of elements for each position within the two arrays and writes the product results to a third array. Since functions can't return arrays, you have to pass in three arrays to your function. All arrays must be the same size.

   *Before*

```
a5[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
b5[] = {10, 10, 10, 10, 10, 10, 10, 10, 10, 10}
```

   *After multiplying the arrays*

```
a5[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 3}
b5[] = {10, 10, 10, 10, 10, 10, 10, 10, 10, 10}
c5[] = {10, 20, 30, 40, 50, 60, 70, 60, 90, 30}
```

6. Write a function that finds the "inverted" product of two arrays and writes the results to a third array. Since functions can't return arrays, you have to pass in three arrays to your function. All arrays must be the same size. The "inverted" product is defined as

```
//pseudo code
//a an array a[1..n]
//b an array b[1..n]
//c an array c[1..n]

c[1] = a[1] * b[n]
c[2] = a[2] * b[n - 1]
....
c[n - 1] = a[n - 1] * b[2]
c[n] = a[n] * b[1]
```

   *Before*

```
a6[] = {1, 2, 3, 4, 5}
b6[] = {6, 7, 8, 9, 10}
```

   *After finding the inverted product of both arrays*

```
a6[] = {1, 2, 3, 4, 5}
b6[] = {6, 7, 8, 9, 10}
c6[] = {10, 18, 24, 28, 30}
```

7. Write a function that reverses the elements in an array. You will have to swap the array elements of the original array parameter.

   *Before*

```
a7[] = {1, 2, 3, 4, 5}
```

*After reversing an array*

```
a7[] = {5, 4, 3, 2, 1}
```

8. Write a function that generates a random array of 10 numbers that are less than **50**. Please output the array you generate. This should be array `a8[]`.

   The following code produces 3 random numbers:

```c
#include <stdio.h>
#include <time.h>
#include <stdlib.h>

int get_random_int(int n);

int main(void)
{
    int p, q, r;
    /* only need to call this once
           to seed the random number generator */
    srand(time(NULL));

    p = get_random_int(16);
    q = get_random_int(1000);
    r = get_random_int(87);

    printf("%d, %d, %d\n", p, q, r);
    return 0;
}



/**
 * generates a random integer between [0,n)
 * @param n the upper bound
 * @return a random integer between [0, n)
 */
int get_random_int(int n)
{
    return rand() % n;
}
```

## 3.2 Getting Help

If you need help with this Lab, please go to tutoring offered weekdays in Cramer 213 and weekends online via discord. Please also ask questions in class to the TA's or instructor. These concepts are important for future labs and exams, so if you don't understand them now it is important to get help! Take advantage of your resources!!

## Submitting

You should submit your code as a tarball file that contains all the exercise files for this lab. The submitted file should be named (**note the lowercase**)

`cse113_groupNUMBER_prelab5.tar.gz`

Replace the `NUMBER` with your assigned group number and have one group member upload the tarball to Canvas before the due date.

Only one person in your group needs to upload your group's tarball to Canvas. If, upon being uploaded, all members of your group do not see that the file has been submitted, please contact the instructor or head TA as soon as possible.

**Upload your `.tar.gz` file to Canvas.**

## List of Files to Submit



Exercises start on page 1.