Tutorial 6

- Arrays.
- Pointer arithmetic.
Arrays

They can be used to store a **fixed number** of elements of the **same** type.

Example of array syntax:

```c
int my_array[3] = { 1, 2, 3 };
int x = my_array[0]; // x = 1
```
Array Initialization

There are several ways to define an array:

```c
int a[3]; // array is not initialized, but it's defined

int b[3] = { 1, 2, 3 }; // array is initialized

int d[3] = {0}; // array of length 3, filled with zeros

int e[8] = { 7, 4, 1 }; // {7, 4, 1, 0, 0, 0, 0, 0}
```
CQ 1:

Which code snippet has an error when placed in the indicated spot?

```c
int arr[10] = { 0 };
// ...
for(/* INSERT ANSWER HERE */) {
    arr[i]++;
}
```

A  int i = 0; i < 10; i++
B  int i = 9; i > -1; i--
C  int i = 9; i != 0; i -= 2
D  int i = 0; i != 10; i += 2
Certain arithmetic operations can be performed on pointers. An integer can be **added or subtracted** to a pointer, and pointers of the same type can be **subtracted** from one another.

```c
int a[10];
int *p = a;  // a is a pointer to first element
int *q = &a[9];  // address of 10th element
q = a + 9;  // equivalent
a[2] = q - p;  // set the value of 3rd element as 9
q = p + 1;  // now q == &a[1]
```

**Addition of pointers is not allowed.**
CQ 2:

```c
int a[3] = { 1 };
int *p = a;
int *q = p + 2;
p += q - p;
++(*q);
```

What is the value of *p after the above code executes?

A 0
B 1
C 2
D -1
E undefined
Exercise: Reverse

// reverse_array(arr, len) reverses the contents of arr
// requires: arr is an array with length len
// effects: modifies arr
Exercise: Subarray

// subarray(a, a_len, b, b_len) determines if b is a subarray of a
// requires: a, b are valid arrays of length a_len, b_len
//           1 <= b_len <= a_len