Pointer arithmetic

Pointer arithmetic is another way to traverse through an array. Again, it is much more common in the C world of programmers. Most faculty at UWB don't want to see pointer arithmetic in your coding.

To demonstrate, declare an int array:

```
int a[10];
```

As you know an array's value is its pointer, so you can create an alias name:

```
int* p = a;
```

Now `a` and `p` are the exact same thing. You can use the standard array notation and refer to a[5] or p[5].

But you can also use `p` to traverse the array by using ++p. In fact, any arithmetic works. Referencing p+2 gets you to p[2]. In that case, the `p` variable isn't incremented and still points to the start of the array. To change p, do something like ++p or p+=2 .

A typical time programmers use this is with char arrays. Consider the declaration and the code to traverse the array. The last element of a C/C++ char array is the NULL character, '\0' . This code simply displays char by char the array contents:

```
char s[] = "hello world";
char* ptr = s;

// no initialization in for loop
for(; *ptr != '\0'; ptr++) {
   cout << *ptr;
}
```

Yet another way to declare a pointer to the beginning of the array is to use the address of a[0]:

```
int* q = &a[0];
```

Note that you can increment any pointer variable, but the operation is meaningless unless you use it on an array.

To make your head hurt, see if you can get correct output for this sample pointer arithmetic program.