Arrays & Const

Arrays passed to a function act as if the array was passed by reference. That can be dangerous, because the function may inadvertently modify the caller's argument.

```
1  for (size_t i = 0; i < len; ++i)
2  {
3      sum += a[i];
4      a[i] = sum;
5  }
```

This function was **intended to sum all the elements** in an array. If you were distracted and inadvertently used an assignment operator instead of the comparison operator, as on line 4, the function would still produce the correct sum, but mistakenly **destroy the values in the array passed to it**.

Not a good thing. To fix this, use the same technique you used for pass-by-reference:

- If a function **intends** to modify the array (initialization, shifting, sorting, etc.) then **do not use** `const` in front of the formal parameter. (Since you are passing by address, **you will never use** `&`.)
- If a function **does not intend to modify the array** (counting, summing, printing, etc.) then **always** use `const` in front of the parameter.

```
double average(const int a[], size_t len);
```