Pointers, Parenthesis and Math

Assume that *p* is a pointer variable.

Expression	Memory/Address	Value at Address
`p`	Yes	No
`*p`	No	Yes
`*p++`	Incremented after value is read	Unchanged
`*(p++)`	Incremented after value is read	Unchanged
`(*p)++`	Unchanged	Incremented after it's used
`*++p`	Incremented before value is read	Unchanged
`*(++p)`	Incremented before value is read	Unchanged
`++*p`	Unchanged	Incremented before it's used
`++(*p)`	Unchanged	Incremented before it's used
`p*++`	Not a pointer	Not a pointer
`p++*`	Not a pointer	Not a pointer

The ++ operator is used above for illustration purposes.

Tip: Use parenthesis to isolate part of the pointer problem and the answer will always work out the way you intended.

Pointers and Array Brackets

Array Notation	Pointer Equivalent
`array[0]`	`*a`
`array[1]`	`*(a+1)`
`array[2]`	`*(a+2)`
`array[3]`	`*(a+3)`
`array[x]`	`*(a+x)`

Pointers to Pointers (** Notation)

The `**` notation indicates the address of a pointer. Assume `**b` is a pointer. Therefore, if `*buffer` is a pointer:

```
**b = &buffer;
```

Assume that *array* is initialized as: `char *array[n];`

Expression	What It Is	Data Type
`*array[]`	An array of pointers	Array
`**array`	An array of pointers	Array (pointer-pointer)
`array+1`	An address	Pointer
`*(array+1)`	Contents of address, what lives there	String/pointer
`*(*array+`*n*`)`	Character *n* of the string at `*array`	Character