The identity of an operation

The box to the right defines three variables, `b`, `s`, and `p`. For example, if `b` contains the values {2, 3, 2}, `s` = 7, and `p` = 12.

Suppose `b` is empty. Then `s` = 0 and `p` = 1. The big question then arises: why is the *product* of an empty bag 1? This little note answers that question and talks about the identity of an operation.

`b` is a bag of integers. `s` is the sum of the integers in `b`. `p` is the product of the integers in `b`.

Why 1 is the product of an empty bag:

Suppose `b` contains {2, 3, 2}, so `s` = 7 and `p` = 12. Suppose we want to insert the value 5 in `b` but keep all definitions true, we would execute:

Add 5 to `b`;
`s=` `s` + 5;
`p=` `p` * 5;

That same sequence should be used to add 5 to `b` no matter what `b` is. Suppose bag `b` is empty and we add 5 to it. Therefore, we want to set `p` to 5. What value should be in `p` when the bag is empty to that execution of

`p=` `p` * 5;

sets `p` to 5? Obviously, `p` must be 1. That is why the product of an empty bag is defined to be 1.

The identity of a binary operation

We define the identity of common operators. The generalization to any binary operator is clear, although not all binary operators have identities:

0 is the identity of + because	0 + `x` = `x`
1 is the identity of * because	1 + x = x
false is the identity of || because	**false** || `c` = `c`
true is the identity of && because	**true** && `c` = `c`

For any binary operator **o** with an identity, **o** applied to the empty bag is the identity of **o**.

The sum of an empty bag is 0.
The product of an empty bag is 1.
The disjunction (“or” ||) of an empty bag is **false**
The conjunction (“and” &&) of an empty bag is **true**.