The new `deriv` procedure uses a table to store the operation to perform for derivatives besides the base cases of constants or the variable itself. The operation to perform is `'deriv` and the "type tag" is the symbolic operator. The reason we can't assimilate the predicates `number?` and `variable?` into the data-directed dispatch is because there is because of the way the expressions that it expects are represented. As base cases, `number?` and `variable?` operate on expressions that are symbols, not lists of symbols. This is crucial for the assimilation of a predicate into the data-directed dispatch, as the `car` of a list expression (`operator`) is used to identify which operation to perform (`'+` for `sum?` and `'*` for `product?` are the tags used in this data-directed dispatch). The procedures `operator` and `operands` cannot be used for expressions that are meant for `number?` and `variable?` as they are not pairs.

When indexing the procedures the opposite way (using `'deriv` as the tag and the operator as the operation to perform), the only change that needs to be made is reversing the order of the first two arguments in each `put` procedure. As long as the order is changed, the `operator` tag will still match the correct operation in the table, since the operator will be the same symbol extracted from the list and the type will be `'deriv`. This works because <*op*> and <*type*> are simply used as an index, so as long as they align with the unique identifier, whether or not they refer to the intended part (operator or type) is irrelevant in this implementation. The initial way of indexing, however, would make it clearer the intended operation and type according to the "documentation" given by the text.