COMPILERS QUIZ 2: EXTENDING FORTH

Add the following features to your forth interpreter.

1 Booleans

Introduce new data values true and false that will push the value into stack. Implement operations not, and, or, xor.

get should read true and false and put it on stack. put outputs true and false to screen when they appear on top of stack. Note that the Boolean true and the string "true" should be distinguished by get and put using double quotes.

The following program outputs false to screen.

```
true false and put
```

2 Comparisons

Implement words `=, <, >, <=, >=, !=' for comparing numbers.

The following program outputs true to screen.

```
45 44 > 45 45 < or put
```

3 Lists

Add list values to the language.

Implement words [and] to create lists. These words are special compared to words so far. The general form is [<program>]. Suppose the stack is S at [and the stack is x1 x2 xn S at]. Then, after], the stack is ([x1, x2, , xn]) S. That is, a list followed by the original stack. It is an error for the <program> to attempt to pop or access values from S.

The following are all ways to create [1, 2, 3] on stack.

```
[ 1 2 3 ]
[ 1 2 1 2 add ]
[ 1 2 3 4 5 6 pop pop pop ]
```

The following lists should cause execution to error.

```
[ pop ]
[ 1 2 pop pop pop 1 2 3 ]
```

Nested lists are allowed. For python list `[1, 2, [3, 4]]', do:

```
[ 1 2 [ 3 4 ] ]
```

Implement a word nth that takes n xs S to xs[n] S. Both the index and list are popped from the stack.

The following prints 3 and 3.

```
[ 1 2 3 ] 2 nth put
[ 1 2 [ 3 4 ] ] 2 nth 0 nth put
```

Implement a word spread that takes a list at the top of the stack and puts individual elements into the stack.

The following computes 1 + 2 * 3 = 7.

```
[ 1 2 3 ] spread * + put
```