```
CODE> a1 y rtn
CODE> a0 x rtn
CODE> b0 z rtn
CODE> ; rtn
: b b0 ;
: a a0 a1 ;
: test a b ;

Ok> test

                                                                   return stack           operations

    (execute test)                             save a              a -                    . Push(first word)
        get a                                                      -                      . Pop
        test_docolon(exectue a)                save b              b -                    . Push(next word)
            a_docolon(execute a0)              save a1             b a1 -                 . Push(next word)
                a0_docode(run x)
                a0_docode(run rtn(x)  == get a1)                   b -                    . Pop
            a_docolon(execute a1)              save exit(a)        b ;a -                 . Push(next word)
                a1_docode(run y)
                a1_docode(run rtn(y)  == get ;)                                           . Pop
            a_docolon(execute ;a)                                  b -                    . Push(next word)
                ;a_docode(run rtn(;a) == get b)                    -                      . Pop
        test_docolon(execute b)                save ;test          ;test -                . Push(next word)
            b_docolon(execute b0)              save ;b             ;test ;b -             . Push(next word)
                b0_docode(run z)                                   ;test ;b -             .
                b0_docode(run rtn(z)  == get ;b)                   ;test ;b -             . Pop
            b_docolon(execute ;b)                                  ;test                  . Push(next word)
                ;b_docode(run rtn(;b) == get ;test)                -                      . Pop
        test_docolon(execute ;(test))                                                     . Push(next word)
            ;test_docode(run rtn(;test) == get stack empty)        -                      . Pop

    test == done

Notes:
        _docolon    push <next> item to execute on the stack
                    run _docolon on <this> item
                        continue (_docolons)*n nesting into the word
                        when a _docode is reached
                            do the <code>
                            do the <rtn>
                                PopR readies for repeat of _docolon until stack is empty
```