```
false 0 ; true -1 ;
off ( a -- ) true swap ! ; on ( a -- ) false swap ! ;
pad ( -- a ) temporary space Vhere @ 256 + ;
The number conversion routines A and B address registers.
hex 16 Base ! ; decimal 10 Base ! ; octal 8 Base ! ;
8 Infd 8 Outfd
type ( a n -- ) Outfd @ -rot ( fd a n ) syswrite drop ;
negate ( n -- -1*n ) not 1 + ;
abs ( n -- |n| ) test -if negate then ;
asciise ( u1 -- u2 ) converts a digit u1 to an ascii value
    10 - -if 1 + 57 ascii code of 9 + ; then
    97 ascii code of a + ;
hold ( c -- ) put c in the ascii string c!- ;
sign ( n -- ) test -if 45 ascii code of - hold ; then drop ;
<# start number conversion 48 ascii code of 0 pad dup A! dup B! c! ;
#> ( -- a n ) A 1 + B A - ;
# ( u1 -- u2 ) write a single digit in ascii at the address in A
    Base @ /mod asciise c!- ;
#s ( u1 -- ) convert u1 to an ascii string at the adress in A
    begin # until ;
. ( n -- ) A push B push dup abs <# #s sign #> type pop B! pop A! ;
write ( a n -- ) Outfd @ -rot ( fd a n ) syswrite drop ;
swrite ( counted_string -- ) Outfd @ swap count syswrite drop ;
newline writes a newline to Outfd   swrite ;
factorial' ( result input -- result*input input-1 )
    dup 1 - push * ( result*input )
    pop ( result*input input-1 ) ;
factorial ( result input -- result*input input-1 )
    test -if drop ; then if factorial' factorial ; then drop ;
main 1 4 factorial 1 Outfd ! . newline ;
```