```
to number x y ()
to class x y ()
to vector x y ()
to atom x y ()
to string x y ()
to false x y ()
'Dont edit above here!'
to printing (CODE 38). printing 0.
to print (%.?("unprintable print) CODE 0)
to # (:#)
print #number. print #atom. print #vector. print #string.
print #false. print #class.

to : (CODE 18)
to % (CODE 17)
to ! (CODE 13)
to " (CODE 9)

"(TITLE USER DO SIZE CODE SELF or and mod rem
 false print chars ref
 ".,/;:-[]~|↑!#$%}←={}*+?<>)

to PUT x y z (:#x. :y. :z. CODE 12)
PUT atom "DO "(CODE 29
        %is?(%atom?() :". !false)
        %print?(disp←SELF chars) )
PUT false "DO "(CODE 11
        %is?(%false?(!1) :".)
        %print?("false print) )
PUT number "DO "(CODE 4
        %is?(%number?() :". !false)
        %print?(disp←nprint SELF 10)
        %base?(!nprint SELF :x) )
PUT vector "DO "(CODE 3
        %is?(%vector?() :". !false)
        %print?(disp←40. for x to SELF length
                (disp←32. SELF[x] print). disp←41) )
PUT string "DO "(CODE 3
        %is?(%string?() :". !false)
        %print?(disp←39. disp←SELF. disp←39) )
PUT USER "TITLE "USER

to repeat x (:#x. CODE 1)
to done x (%with?(:x. CODE 25) CODE 25)
to again (CODE 6)
to ev (repeat (read eval print cr.))
"eval←"eval
to read (CODE 2)
to nprint r n i s :: pname (%knows?(ev):n. :r. "i←18.
        (n<0?("s←1. "n←0-n)"s←0)
        repeat (pname["i←i-1] ← 48+n-r*"n←n/r. n=0?(done))
        (s=1?(pname["i←i-1]←45)). !sscan pname[i to 17])
nprint knows
"pname←string 17. done

to right (CODE 21)
to go (CODE 22)
to TURTX x (:x. CODE 23)
to ink x (:x. CODE 32 x)
to penup (TURTX 0)
to pendn (TURTX 1)
```

```
to home (TURTX 2)
to up (TURTX 3)
to erase (TURTX 4)
to black (ink 0-1)
to white (ink 0)
to turtlestate : pen ink dir x xf y yf dx dxf dy dyf (
        isnew?(0 CODE 39)
        %pause?(0 CODE 39)
        %resume?(1 CODE 39)
        %knows?(!(:".)eval) )

to filin x (:x. CODE 16 x)
to STOP (CODE 8)
to GET x y (:#x. :y. CODE 28)
to mem x y (:x. CODE 26)
to - (0-:)
to trace (CODE 34)
to mouse x (:x. CODE 35)
to core ((mem 59)-mem 58)
to ov (CODE 15)
to isnew (CODE 5)
to null x (:x. 1 CODE 37)
to error (print :. CODE 14)
to eq x y (:x. :y. 1 CODE 33)
to kbd (CODE 20)
to disp x i (%←.
        :x is string?(for i to x length (TTY←x[i]))
        TTY←x)
to TTY (CODE 20)
to dsoff (mem 272←0)
to dson (mem 272 ← mem 62)
to apply x y (:#x. %to?(:y. CODE 10) CODE 10)
to cr (disp←13)
to wboot (CODE 27)
to is (:". !false)
to nil x (#x)

to if exp (:exp?(%then?(:exp. %else?(:". exp)exp)error "(no then!))
        %then?(:".   %else?(:exp) false)error "(no then!))
to for x step stop var start exp (
        :"var. (%=?(:start.)"start←1.)
        (%to?(:stop.)"stop←start.)
        (%by?(:step.)"step←1.)
        %do. :#exp. CODE 24)
to do N exp (:N. :#exp. for N to N do (exp eval.))

to dragon i (:i=0?(go 5)
        i>0?(dragon i-1. right 90. dragon 1-i)
        dragon -i+1. right -90. dragon i+1)

to edit func t char :: ed1 edsearch edmatch eddel edins (
        %knows?(ev)
        "char←0. :#func. "t←GET func "DO.
        null t ? (error"(no code)) PUT func "DO ed1 t. "done)
edit knows
to ed1 ptr n back(:ptr. "back←false. repeat(
        (char=120?(done with ptr) char=63?() (char=0?()cr). for n to ptr length-1
                (disp←32.ptr[n] is vector?("$ print)ptr[n] print)) "n←0. cr.
        repeat (47<("char←disp←kbd)<58 ? ("n←(char-48)+10*n)
                char=45?("back←1)done)
        (char=115 ? (edsearch)  )
```

```
        (back?("n←ptr length - n."back←false))
        char=108 ? (done with ptr)
        char=120 ? (done with ptr)
        (n<1?("char←63))
        char=101 ? (ptr[n]←ed1 ptr[n])
        char=114 ? (ptr[n]←read[1])
        "n←n-1
        char=105 ? (edins)
        char=100 ? (eddel)
        disp←"char←63) )
to edsearch target i ("target←read[1]. (n=0?("n←1)).
        back?(for i=ptr length to 1 by -1 (edmatch))
        for i to ptr length (edmatch) )
to edmatch (eq target (ptr[i] is vector?("$)ptr[i]) ?
        (0="n←n-1?(cr. "n←i print. "char←disp←kbd. "back←false. done)) )
to edins t1 t2 ("t1←read. "t2←vector t1 length + ptr length-1.
        do n (t2[N]←ptr[N]).
        do t1 length-1 (t2[n+N]←t1[N]).
        do ptr length-n (t2[n+t1 length+N-1]←ptr[N+n]).
        "ptr←t2.)
to eddel t1 t2 ("t2←vector ptr length - "t1←(disp←kbd)-48.
        do n (t2[N]←ptr[N]).
        do ptr length-t1+n (t2[N+n]←ptr[N+n+t1]).
        "ptr←t2.)
done
to addto func v w (:#func. :w. "v←GET func "DO.
        null v?(error "(no code)). PUT func "DO catcode v w.)
to catcode v w x (:v. :w. "x←sscan v[1 to v length+w length -1].
        !sscan x[v length to x length]←w[1 to w length])

to sscan op byte s lb ub s2 lb2 ub2 (
"lb2 ← "ub2 ← 1.
:#s. %[. :lb. %to. :ub. %].
%find? ("op ← (%first?(1) %last?(2) 1) + (%non?(2) 0). :byte. CODE 40)
%←?  (%all? (:byte. "op←0. CODE 40)
       :#s2.%[:lb2.%to.:ub2.%]. "byte ← 0. "op ← 5. CODE 40)
"op ← 6. "byte ← 0. "ub2 ← ub+1-lb.
"s2 ← (s is string?(string ub2)vector ub2).  CODE 40)

to cons : head tail (
        %tail?(%←?(!:tail)!tail)
        %head?(%←?(!:head)!head)
        (isnew?(:head. :tail))
        %print?(disp←40. head print. disp←46. tail print. disp←41))
to sequence : str ptr (
        %scan?(CODE 41?(!rpar))
        %pos?(%←?(!:ptr)!ptr)
        %fill?(for ptr to str length
                (10=str[ptr]←disp←kbd?(done))"ptr←0)
        isnew?(:str. "ptr←0) )

to reed v s : : ownv lpar rpar subreed growreed (%knows?(ev)
        "v←ownv. "s←v length. !subreed 1)
reed knows
to subreed i j t ("j←:i.
        repeat("t←reader scan.
                rpar=t?(v[j]←nil. done with sscan v[i to j])
                lpar=t?(v[j]←subreed j. s < "j←j+1?(growreed))
                v[j]←t. s < "j←j+1?(growreed) ))
to growreed ("v←sscan v[1 to "s←s+ (200>core?(50)s)] )
"ownv←vector 200.
```

```
"reader←sequence string 2. reader fill.
()  "lpar←reader scan. "rpar←reader scan.
done

printing 1 STOP "(2/5 Smalltalk)
←
```

```
printing 0.
'file maintenance.    DP0:litdisplay  DP1:litdisplay'
to D (print :". ev)

(to dispframe input : winx winwd winy winht frmx frmwd frmy frmht buf mode
        last toplen lstln charx chary reply : init clearit scroll append (
% ← ?(:input is string ?(append. "mode←0. CODE 51
                reply>0?(SELF scroll !input)!input)
        input=8 ?((mark<last?("last←last-1)).
                buf[last+1]←0. "mode←input. CODE 51 !input)
        buf["last←last+1]←"mode←input.  CODE 51.
                reply>0?(SELF scroll !input)!input)
%scroll?(CODE 54 scroll. "mode←257. CODE 51 "mode←256. CODE 51
        reply>0?(SELF scroll))
%putmark?("mark←last).
%getmark?(mark<0?(!last)!mark).
%clear?(clearit CODE 51)
%knows?(ev)
%param?(!(:".)eval)
isnew ?(apply init) ))
dispframe knows
to init ("winx←:frmx. "winwd←:frmwd. "chary←"winy←:frmy.
        "winht←:frmht. :buf. "lstln←1.
        "mode←"mark←"last←"toplen←"charx←"reply←0)
to clearit ("last←0. "lstln←1. "chary←winy.
        sscan buf[1 to 2000] ← all 0.  "mode←257.)
to scroll end ("end←buf length.
        sscan buf[1 to end] ← buf[toplen to end].
        "last ← last - toplen-1. "mark ← mark-toplen-1)
to append end ("end←buf length.
        sscan buf[last+1 to end] ←input [1 to end].
        end<"last←last+input length?("last←end) )
done
to rdisp (disp←45. disp←62. disp putmark.
        repeat (10=disp←kbd?(done)) disp←13.
        reader pos ← disp getmark)
to start(erase."disp←dispframe 0 512 512 160 string 520. disp clear.
        DISK. "reader←sequence disp param (buf).
        "fool←#read. to read (rdisp. !reed) "Hello)

printing 1
```

```
printing 0
edit string
-i      %=?     (SELF length=:y length? (for x to SELF length
                        (SELF[x]=y[x]? () !false)) !false)
-i      %fill?  (disp←42. for x to SELF length (SELF[x]←disp←kbd))
]

edit number
-i      %mod?   (!SELF-:x*SELF/x)
-i      %max?   (SELF<:x? (!x) !SELF)
]

(to file : bytec status dirty leader curadr sn1 sn2 version
        backp nextp pagen numch 18 sadr fname dirinst bitinst (

        %next?  (%word? (%←? (7 CODE 50) 6 CODE 50)
                 %char.  %←? (1 CODE 50) 0 CODE 50 )
        %eof?   (10 CODE 50? () !false)
        %set?   (%to. %eof? (15 CODE 50)
                 %write? (%next? (%page. 17 CODE 50)
                        %page? (13 CODE 50)
                        %byte? (14 CODE 50) )
                 %read. %next? (%page. 16 CODE 50)
                        %page? (4 CODE 50)
                        %byte? (5 CODE 50) )

        %rewind?        (11 CODE 50)
        %close? (12 CODE 50)
        %is?    (%file? () :". !false)
        %skipnext?      ("bytec←bytec+:.)
        %param? (!(:".) eval)
        %ddt?   (ev)
        %load?  (8 CODE 50? (!false))
        %save?  (9 CODE 50? (!false)) ))
edit file
-i      isnew?  ((:fname=DIRNAME? ("dirinst←SELF) "dirinst←DIRECT.
                fname=BITNAME? ("bitinst←SELF) "bitinst←BTABLE)
                "sadr ← string 512.
                %new?   (3 CODE 50? ("sadr←0. !false) !SELF)
                %old.    2 CODE 50? ("sadr←0. !false) !SELF )
]

to DIR ch (DIRECT rewind. repeat (
        DIRECT eof? (done)
        (("ch←DIRECT next word)/1024)=0? (DIRECT skipnext((ch-1)max 0)mod 512)
        DIRECT skipnext 10.
        for i to "ch←DIRECT next (disp←DIRECT next). cr.
        (ch mod 2)=0?   (DIRECT next) ))

to [ n (radix 8 ((disp←32.:n<0? (disp←49.print 32767+n+1) print n)))
to DISK ("DIRECT ← file DIRNAME old.
        "BTABLE ← file BITNAME old.
        DIRECT param ("bitinst ← BTABLE) )

("DIRNAME ← string 7) fill
SYSDIR.
("BITNAME ← string 9) fill
SYS.STAT.
printing 1
```

```
printing 0.
to table token i : len nam val (
        %enter?(:token.
                (0="i←sscan nam[1 to 1000] find first token?
                        (len=nam length?(error "(table full))"i←"len←len+1))
                nam[i]←token. val[i]←fptr)
        %lookup?(:token. 0="i←sscan nam[1 to len] find first token?
                (!false)!val[i])
        %knows?(ev)
        isnew?("nam←vector :token. "val←vector token. "len←0)
        !sscan nam[1 to len])
"fptr←0.
"defs←table 150
"fool←#to. to to x (CODE 19 defs enter x. x)
"reader←sequence string 50
to read (disp←62. reader fill. !reed)
printing 1.
PUT USER "DO "(cr. read eval print.)
v
```

```
printing 0
addto string "(
        %=?     (SELF length=:y length? (for x to SELF length
                        (SELF[x]=y[x]? () !false)) !false)
        %fill?  (disp←42. for x to SELF length (SELF[x]←disp←kbd)) )"

addto number "(
        %mod?   (!SELF-:x*SELF/x)
        %max?   (SELF<:x? (!x) !SELF) )"

(to file : bytec status dirty leader curadr sn1 sn2 version
        backp nextp pagen numch 18 sadr fname dirinst bitinst (
        %next?  (%string? (%←? (17 CODE 50) %into? (16 CODE 50) !false)
                 %word? (%←? (7 CODE 50) 6 CODE 50)
                 %char.  %←? (1 CODE 50) 0 CODE 50 )
        %eof?   (10 CODE 50? () !false)
        %set?   (%to. %eof? (13 CODE 50)
                 %write?        (5 CODE 50)
                 %read. 4 CODE 50 )
        %rewind? (11 CODE 50)
        %close? (12 CODE 50)
        %is?    (%file? (!fname) :". !false)
        %print? (fname print)
        %skipnext?      ("bytec←bytec+:.)
        %param? (!(:".) eval)
        %ddt?   (ev)
        %load?  (RELEASE. 8 CODE 50? (!false))
        %save?  (RELEASE. 9 CODE 50? (!false) (!fname) REOPEN) ))
addto file "(
        %name?  (!fname)
        %size?  (SELF set eof. !pagen)
        %=?     ((eq #SELF :.)? () !false)
        %shorten?       (%to. 14 CODE 50)
        %delete?        (15 CODE 50? (!false) filesopen without #SELF)
        isnew?  ((:fname=DIRNAME? ("dirinst←SELF) "dirinst←DIRECT.
                fname=BITNAME? ("bitinst←SELF) "bitinst←BTABLE)
                "sadr ← string 512.
                %new?   (3 CODE 50? (!false) filesopen←#SELF. !SELF)
                %old.    2 CODE 50? (!false) filesopen←#SELF. !SELF ))"

to DIR ch (DIRECT rewind. repeat (
        DIRECT eof? (done)
        (("ch←DIRECT next word)/1024)=0? (DIRECT skipnext((ch-1)max 0)*2)
        DIRECT skipnext 10.
        for i to "ch←DIRECT next (disp←DIRECT next). cr.
        (ch mod 2)=0?   (DIRECT next) ))

to DISK ("filesopen ← stack 10. "DIRECT ← file DIRNAME old.
        "BTABLE ← file BITNAME old. DIRECT param ("bitinst←BTABLE) )

to stack i x : top vec (isnew? ("top←0. "vec←vector :.)
        %←?     (vec["top←top+1] ← :.)
        %without?       (:x. for i to top (vec[i]=x?
                (for i=i to "top←top-1 (vec[i]←vec[i+1]) done)))
        %length?  (!top)
        %print? (for i to top (vec[i] print. cr))  !vec)
to filestat (filesopen print.)
to REOPEN i (for i to filesopen length (filesopen[i] set 0 0))
to RELEASE i (for i to filesopen length (filesopen[i] close))

("DIRNAME ← string 7) fill
```

```
SYSDIR.
("BITNAME ← string 9) fill
SYS.STAT.
to load dname (file :dname load)
to save x dname (("x←file :dname )?(x save)("x←file dname new)?(x save)
        !false)
printing 1
```

```
printing 0.
'file maintenance.    DP0:litdisplay  DP1:litdisplay'
to D (print :". ev)

(to dispframe input : winx winwd winy winht frmx frmwd frmy frmht buf mode
        last toplen lstln charx chary reply
        : init clearit scroll append translate (
% ← ?(:input is string ?(append. "mode←0. CODE 51
                reply>0?(SELF scroll !input)!input)
        input=8 ?((mark<last?("last←last-1)).
                buf["last+1]←0. "mode←input. CODE 51 !input)
        buf["last←last+1]←"mode←input.  CODE 51.
                reply>0?(SELF scroll !input)!input)
%scroll?(CODE 54 scroll. "mode←257. CODE 51 "mode←256. CODE 51
        reply>0?(SELF scroll))
%putmark?("mark←last).
%getmark?(mark<0?(!last)!mark).
%clear?(clearit CODE 51)
%knows?(ev)
%param?(!(:".)eval)
isnew ?(apply init) ))
dispframe knows
to init ("winx←:frmx. "winwd←:frmwd. "chary←"winy←:frmy.
        "winht←:frmht. :buf. "lstln←1.
        "mode←"mark←"last←"toplen←"charx←"reply←0)
to clearit ("last←0. "lstln←1. "chary←winy.
        sscan buf[1 to 2000] ← all 0.  "mode←257.)
to scroll end ("end←buf length.
        sscan buf[1 to end] ← buf[toplen to end].
        "last ← last - toplen-1. "mark ← mark-toplen-1)
to append end ("end←buf length.
        sscan buf[last+1 to end] ←input [1 to end].
        end<"last←last+input length?("last←end) )
done
to clear x w y h (:x. :w. :y. :h. CODE 53)

to rdisp : : translate (%knows?(ev)
        disp←45. disp←62. DRIBBLE close. disp putmark.
        repeat (012=DRIBBLE next←disp←translate[kbd]?(done)) disp←13.
        reader pos ← disp getmark. clear 0 512 674 32)
rdisp knows
"translate ← string 0377.
for i=001 to 0177(translate[i]←translate[0200+i] ← i)
translate[0200]←translate[0233]← 040. 'controls null and escape'
to translation i j(:i. :j. translate[i]←translate[0200+i]←j)
translation 0020 0010   'SHIFT BS'
translation 0021 0011   'SHIFT TAB'
translation 0037 0040   'SHIFT UP'
translation 0036 0040   'UP'
translation 0035 0040   'SHIFT DOWN'
translation 0034 0040   'DOWN'
translation 0030 0040   'SHIFT SPACE'
translation 0025 0015   'SHIFT RETURN'
translation 0027 0010   'SHIFT DEL->BS'
translation 0177 0010   'DEL->BS'
translation 0032 0040   'SHIFT INS'
translation 0031 0040   'INS'
translation 0023 0040   'SHIFT ESC'
translation 0022 0012   'SHIFT LF'
done
to start dname ( (DIRECT is file?()DISK)
```

```
        (:"dname is atom?("dname←dname chars. "DRIBBLE←file dname old?()
                "DRIBBLE←file dname new?(error "(help!)))
                error "(type start <name>) )
        DRIBBLE set to eof.
        "disp←dispframe 16 496 512 160 string 520. disp clear.
        "reader←sequence disp param (buf).
        "fool←#read. to read (rdisp. !reed) "Hello)
"DIRECT←0
printing 1
```

don't we create dname so we're in bad news?

```
printing 0.

to space n i (for i to :n (disp←32))
to lpar (disp←40)
to rpar (disp←41)

to show func t (
        :#func.
        "t← GET func "DO.
        null t ?(error "(no code))
        pshow t 0.)

to pshow ptr dent i (
        :ptr space :dent. lpar
        for i to ptr length-1
                (ptr [i] is vector ?((i=1 ?() cr.space dent +1)
                                        pshow ptr [i] dent+1.
                                        (i=ptr length-1 ?()
                                        cr.space dent+1) )

                (i=1 ?() space 1)
                ptr[i] print.)
        rpar)

printing 1.
```

```
printing 0.
to table token i : len nam val (
        %enter?(:token.
                (0="i←sscan nam[1 to 1000] find first token?
                        (len=nam length?(error "(table full))"i←"len←len+1))
                nam[i]←token. val[i]←fptr)
        %lookup?(:token. 0="i←sscan nam[1 to len] find first token?
                (!false)!val[i])
        %knows?(ev)
        isnew?("nam←vector :token. "val←vector token. "len←0)
        !sscan nam[1 to len])
"fptr←0.
"defs←table 150
"fool←#to. to to x (CODE 19 defs enter x. x)
"reader←sequence string 50
to read (disp←62. reader fill. !reed)
printing 1.
PUT USER "DO "(cr. read eval print.)
```