```
  lsw:b #10-#10, #11-#10, #12-#10, #13-#10,
    #14-#10, #15-#10, #16-#10, #17-#10,
    #18-#10, #19-#10, #110-#10, #111-#10,
    #112-#10, #113-#10, #114-#10, #115-#10

  $end loader

$begin transfer
  $def chars=r3, lines=r7
  $def terminator=r9,intypes=r10,outtypes=r11

  print:
    s2 = #lp                        /output to printer
    outtypes = x'800'               / which is a file
  trans:
    if terminator = nl
      intypes = intypes&x'8FF'
      outtypes = outtypes&intypes
      if intypes = x'800' and outtypes = intypes
        ac = 'o'+double             /filestore copy
        fcomm
        stop
      finish
      ac = x'FFFF'
    else
      rdec; rsym
    finish
    lines = ac; chars = 0
    refout 1,s2
    cycle                           /for each file
      refin 1,inn1
      cycle
        rsym
        exit if sym < 0
        chars = chars+1
        psym sym
        if sym = nl
          lines = lines-1
          stop if lines = 0
        finish
      repeat
      closin
      inn1 = inn1+20
    repeat if inn1 < outn1
    selout 0
    ac = chars
    pdec; newline
    stop

$end

  $begin ecce (screen editing version)

  /Registers
  $temp r1
$def ac=r0, sym=r1,temp=r1, work=r3,ret=r3
  $def line=r5
  $def pend=r4, rcret=r6            /command input
/     p=r4, min=r6, no=r7           /execution
  $def ci=r8, code=r9, text=r10, num=r11
  $def ti=r12, chain=r13, type=r14, item=r15 /command input
/     lbeg=r12, lend=r13, pp=r14, fp=r15 /execution

  $def c=0, t=2, n=4                /code,text,num fields
$def maxlin=90
  $def sp=\3, not=3

$macro var n,b[2]
  $def n=base
  $redef base=w(#base+b)
```

```
$def c=0, t=2, n=4              /code,text,num fields
$def maxlin=90
$def so=\3, not=3

$macro var n,b[2]
 $def n=base
 $redef base=w(#base+b)
$end
 $def base=save(pb)

var lbeg1                       /*r12*
 var lend1
 var pp1
var fp1                         /*r15*
 var killed                     /*r3*
 var limit
var secfp
 var sin
 var outblocks
var ms
 var ml
 var clim
var mode
 var casing
 var error
var sbeg
 var slim                       /*r15*
 $def in2=b(inout0+4)
$def topcb=out1(pb), botcb=in1(pb), seccb=in2(pb)
 $def fend=slim, cbuff=#base, tlim=nplim(pb)
 $def inf=x'7FFF'


 $/command input routines

 stype:
 /     !"#       $%&'       ()*+       ,-./
   x'0333', x'3233', x'AC03', x'B233'
 /     0123       4567       89:;       <=>?
   x'0000', x'0000', x'0031', x'333D'
 /     @abc       defg       hijk       lmno
   x'9299', x'5749', x'9699', x'9799'
 /     pqrs       tuvw       xyz[       \]^_
   x'9296', x'5559', x'222A', x'DC33'

 routine readsym,ac
   sym = pend
   rsym if sym = 0
   pend = 0
   jump out

 routine number                 /accept number
   jump out if type # 0
   jumps rd1
 routine minus                  /accept minus
   jump out if item # '-'
   code = code-5                /e->@, f->a, m->h
 routine rditem
rd1:type = 1
   cycle
     readsym
     jump out if sym < ' '      /control =>
   repeat if sym = ' '
   sym = sym-32 if sym >= 96    /map lower-case to upper
   item = sym
   sym = sym>>1                 /*sets cc*
   type = stype[sym-16]         /*preserves cc*
   type = type>>4 if \8         /sym was even
   type = type&15
```

```
  jump out if type # 0
  num = item-'0'
  num = 0 if num < 0              /'*'
  cycle
    readsym
    pend = sym
    sym = sym-'9'
    jump out if sym > 0           /not digit  =>
    sym = sym+9
    jump out if sym < 0           /not digit  =>
    num = num<<1; sym = sym+num
    num = num<<2; num = num+sym /10*num+sym-'0'
    pend = 0
  repeat

routine unchain
  cycle
    text = chain
    jump out if text = 0
    chain = t(text)
    t(text) = ci
  repeat if w(text) # 'l'&31
  jump out

routine stack
  ci = ci+6
  jump er6 if ci >= ti
  c(ci-6) = code
  t(ci-6) = text
  n(ci-6) = num
  jump out

edgram:
  b 'F',50,
  '-',4,
  '.',35,
  ' '+128,'.'-1+128,30,
  '.',6,
  '*',13,
  '0'+128,'9'+128,18,
  11,13,
  '\',22,
  11,26,
  '?',26,
  11,26,
  ',',0,
  11,0,
  '.'+1+128,126+128,255,
  11,9,
  '/',119,
  ' '+128,'/'-1+128,44,
  '/',37,
  11,11,
  '/'+1+128,126+128,255,
  11,40,0,
  'D',54,
  11,4,
  'I'+128,'V'+128,59,
  11,4,
  'I',83,
  '.',72,
  '.',67,
  11,11,
  ' '+128,126+128,108,
  11,63,
  '/',255,
  '/',78,
  11,11,
  ' '+128,126+128,108,
  11,74,
```

```
    11,11,
    ' '+128,126+128,108,
    11,63,
    '/',255,
    '/',78,
    11,11,
    ' '+128,126+128,108,
    11,74,
    'S',87,
    11,61,
    'E',93,
    '-',11,
    11,11,
    'M',97,
    11,89,
    'B'+128,'W'+128,102,
    11,11,
    '(',110,
    pcall,0,
    ')',114,
    11,11,
    '[',118,
    11,104,
    ']',118,
    11,11,
    0,
    '0'+128,'9'+128,255,
    '0'+128,'9'+128,4,
    11,122
$def edcomm=*-#edgram
    b '0'+128,'9'+128,edcomm+8,
    '0'+128,'9'+128,edcomm+15,
    11,edcomm+3,
    '%',edcomm+18,
    'A'+128,'Z'+128,255,
    pcall,0,
    nl,255,
    0,
    rt,edcomm+21,
    0,
    '.'+128,'@'+128,edcomm+25,
    0,
    'J',edcomm+28,
    0,
    'w',edcomm+31,
    0,
    'o',edcomm+13,
    0
 $def any=*-#edgram
 b nl+128, 127+128, 255,
    0
 prom:b 3,'>',soh,edcomm
 erprom:b 3,'!',soh,edcomm
anyprom:b 2,soh,any
 routine readco,rcret
 c1:
    ci = clim; pend = 0
    gram(pb) = #edgram
    cycle
      temp = error
      prompt #prom(temp)
      error = 0
      rsym
      jump out if sym = rt and ci # 0 /->repeat last command
    repeat if sym <= ' '
    code = -1; pend = sym
    code = '1'&31 if pend = ';'
    code = 'r'&31 if pend = ':'
    code = 'h'&31 if pend = '.' /m-
    code = 'm'&31 if pend = '/'
```

```
  code = 'e'&31 if pend = 'j'
  code = pend&31 if pend = '@' or pend = 'w' or pend = 'o'
  jump out if code >= 0
  rditem                          /read first item
  if item = '%'
    readsym
    code = sym&95
    pend = -1
    jump out if code = 'C' or code = 'D' or code = 'S'
    code = code-'U'
    code = 32 if code # 0
    casing = code
    jump c1
  finish
  if type = 0 and pend = nl and ci # 0
    n(ci-6) = num
    pend = 0
    jump out
  finish
  ci = chuff; ti = tlim-1
  chain = 0
cget:
  code = item&31; num = 1
  text = tsw[type]
  rditem if type >= 4
  temp = text; text = 0
  jump c00(temp)
c00:c20: c30:
  jump er2
c10:
  abandon if sym < 0
  jump c1
c40: /Find
  minus
  num = 0 if type # 0
c50: /+Delete, Traverse, Uncover, (Verify)
  code = code+(num<<5)
  number
  num = 0                         /as indicator
c60: /+Insert, Substitute
/  jump er4 if type # 3
  text = ti
  cycle
    readsym
    if sym = nl
      pend = sym
      sym = item
    finish
    exit if sym = item
    sym = sym-casing if sym >= 96 and num = 0
    b(ti) = sym; ti = ti-1
    jump er6 if ti = ci
  repeat
/  jump er4 if ti=text and num = 0
  b(ti) = 0; ti = ti-1
  num = 1                         /restore default
  rditem
  jumps c90
c70: /Move, Erase
  minus
c90: /Get, Kill, etc
/  jump er1 if type = 3
  jump c121
c100: /Open bracket
  code = '['&31
  jumps c111
c110: /Comma
  code = ','&31
  num = 0
/  rditem if type = 1             /permit line break
```

```
c110: /Comma
   code = '^'&31
   num = 0
/  rditem if type = 1          /permit line break
c111:
   text = chain
   chain = ci
   jumps cput
c120: /Close bracket
   unchain
   jump er5 if text = 0
   code = '1'&31
   n(text) = num
   text = text+6
c121:number
cput:c130:
   stack
   jump caet if type # 1
   cycle
     unchain
   repeat if text # 0
   code = '1'&31; text = cbuff; num = 1
   stack
   clim = ci
   pend = 0
   jump out

/recognition errors
er1:
/  space
/  psym code+64
er2:
/  code = item-64
/  jumps er5
er4:
/  ptext; b' TEXT FOR',0
/  code = code&31
er5:
/  space
/  psym code+64
/  jumps er9
er6:
/  ptext; b'SIZE',0
er9:
/  psym '?'
/  newline
   clim = 0 if ci # cbuff
/  ignore
   error = #erprom-#prom
   jump c1

tsw:b #c00-#c00, #c10-#c00, #c20-#c00, #c30-#c00,
   #c40-#c00, #c50-#c00, #c60-#c00, #c70-#c00,
   0, #c90-#c00, #c100-#c00, #c110-#c00,
   #c120-#c00, #c130-#c00, #c20-#c00, #c20-#c00

$/* ti,chain,type,insym dropped *
$def p=r4, min=r6, no=w(r7)
$def lbeg=r12, lend=r13, pp=r14, fp=r15

/Set file pointer to start of line
routine lstar,ac
   cycle
     jump out if pp = lbeg
     pp = pp-1; fp = fp-1
     b(fp) = b(pp)
```

```
  repeat

/Set file pointer to end of line
routine rstar,ac
  cycle
    jump out if fp = lend
    b(pp) = b(fp)
    pp = pp+1; fp = fp+1
  repeat

/Write disk block
 /p: cb
 /temp: no of bytes
routine put,ac
  temp = 0 if temp = 512
  temp = temp+s                 /data there
  pointer(p) = pointer(p)+temp
  service p
  jump out if devinf(p) > 0 and pointer(p)&(\bmask) = 0
  abandon

 /Expel one block from start of store
/  shift up rest of top info (ie up to pp)
 /  NB lbeg >= sbeg+512 (except when closing)
nroutine puttop
  temp = lbeg-sbeg              /removeable info
  jump edstop if temp <= 0
  temp = 512 if temp > 512      /restricted to one block
  pp = pp-temp; lbeg = lbeg-temp /adjust
  p = topcb
  put if p >= dcb0
  outblocks = outblocks+1
  p = sbeg                      /Shift up
  while p < pp
    w(p) = w(p+512)
    p = p+2
  repeat
  return

 /Bring in next block of file if any
nroutine slave
  if pp > fp-512
    return if lbeg < sbeg+512 /(overlength lines)
    puttop
  finish
  return if botcb < dcb0 or devinf(botcb) <= 0
                                /fall through
 /Shift up bottom info (from fp)
 /  read one block to end of store
/  NB fp-pp >= 512, lend = slim
nroutine getbot
  push lbeg; lbeg = pp          /Shift up to free lower buffer
  ms = 0 if fp # ms
  rstar
  p = botcb /Read block
  service p
  lend = pointer(p)             /neg if data there
  pointer(p) = lend&bmask       /keep pointer at buffer start
  lend = lend-s
  lend = slim if lend < 0
  fp = lend
  lstar /bring back
  ms = fp if ms # 0
  pop lbeg
  return
```

```
/Shift down top part of info
/  read one block back from output file to start of store
/  NB outblocks > 0, fp-pp >= 512, lbeg = sbeg
nroutine gettop
  push lend; lend = fp          /Shift down
  lstar
  p = topch                     /Read back block
  service p
  pointer(p) = lbeg
  outblocks = outblocks-1
  lbeg = lbeg+512; pp = lbeg
  rstar
  pop lend
  return

/Expel one block from end of store to temp file
/  shift down bottom info
/  NB lend < slim-512 (except sometimes for %S)
nroutine puthot
  p = hotch
  temp = 512; put               /write block
  fp = fp+512; lend = lend+512
  p = slim                      /Shift down
  while p > fp
    p = p-2
    w(p) = w(p-512)
  repeat
  return

/Display (rest of) line
routine displine,ac
  cycle
    p = fp if p = pp
    if p = slim
      push ac
      temp = lend-fp; push temp
      lend = p
      slave
      p = lend; pop temp; lend = fp+temp
      pop ac
      jump out if p = slim
    finish
    sym = b(p); p = p+1
    jump out if sym < ' '
    psym sym
  repeat

$def vline=min                  /NB

routine update,ac
  jump out if min = inf         /no change
  if line >= 0 and line < vlast /on screen
    push ac
    min = pp if min > pp
    ac = line<<8+min-lbeg+x'2000'
    setcursor
    p = min; vline = line
    cycle
      displine
      clearline
      vline = vline+1
      exit if vline = vlast
      exit if killed = 0 and p # pp
      newline
    repeat
    pop ac
```

```
      putbot if fp < pp+512
      gettop
    finish
    exit if lbeg # pp and b(lbeg-1) < ' '
    lbeg = lbeg-1
  repeat
mp1:
  lend = fp-1
  lstar
  line = line-1
  jump mset if line = -2
  jump out

routine insert
  if pp = fp                    /store full
    push sym
    if lbeg-sbeg >= 512
      puttop
      min = min-512             /(ok if min=inf)
    else
      putbot
    finish
    pop sym
  finish
  min = pp if pp < min
  b(pp) = sym
  pp = pp+1
  jump out

routine verify,r0
  p = fp; work = text
  if lend # fend
    cycle
      sym = b(p)
      sym = sym-casing if sym >= 96
      exit if sym # b(work)
      p = p+1; work = work-1
    repeat
  finish
  jump out if b(work) # 0       /fail (cc non-zero*)
  ms = fp
  p = p-fp; ml = p
  temp = 0                      /(cc zero*)
  jump out

$/ entry points
$macro mmove s,d
s1=s; s2=d; smove
$end

ed:
  outn1 = inn1 if b(outn1) = 0 /no output specified
show:
/  mmove cnam,inn1 if b(inn1) = 0
/  mmove outn1,cnam if b(outn1) # 0 and b(outn1+1) # '.'
  refin bg+8+1,inn1
  refin 2,inn1+20
  refout bg+1,outn1
  selin 0; selout 0
  r3 = 0; r4 = 0                /killed, limit
  r6 = 0; r7 = 0                /sin, outblocks
  r8 = 0; r9 = 0                /ms, ml
  r10 = 0 ; r11 = x'2000' /clim, mode
  r12 = 32; r13 = 0             /casing, error
  r14 = pointer(bg)             /sbeg (&pp)
  r15 = pointer(bg+8)+512       /slim (&fp)
/INTERDATA 70/74/7-16 OPERATING SYSTEM
/Department of Computer Science
/Edinburgh University
/ Editing station version from LEGOS:22
```

```
    r10 = 0 ; r11 = x'2000' /clim, mode
    r12 = 32; r13 = 0           /casing, error
    r14 = pointer(bq)           /sheq (&pp)
    r15 = pointer(bq+8)+512     /slim (&fp)
    r5 = r15                    /secfp
    stm r3,killed
    attend
    min = inf
    lend = fp-1
    move
new:
    temp = 0
ernew:
    error = temp
    update
    display
    stm lbeg,lbeg1
    readco
    lm lbeg,lbeg1
    r7 = #lno
    min = inf
    if pend # 0
      if pend < 0
        jump edfin if code = 'C'
/       line = 999 if code = 'D'
        if code = 'S'
          switch
          min = pp; killed = vlines
        finish
        jump new
      finish
      num = 1
      jump rep
    finish
    ci = chuff
nxt:
    jump new if ci = clim       /end of command
    code = c(ci)&31
    text = t(ci)
    num = n(ci)
    ci = ci+6
rep:
    jump x0(xsw[code]<<1)

oka:
    min = pp if pp < min
ok:
    num = num-1
    jump nxt if num = 0
    jump rep

m50:
    if c(ci) = '\'&31 or c(ci) = '?'&31
      ci = ci+6
      jump nxt
    finish
    while c(ci) <= 'I'&31
      ci = t(ci) if c(ci) = '['&31
      ci = ci+6
    repeat
    num = n(ci)
    ci = ci+6
lno:
    jump nxt if num <= 0
    jump m50 if ci # clim
/Execution error
/     selout 0
/     ptext: b'FAILURE: ',0
/     if code <= 'A'&31 or code = 'H'&31 /@,a,h
/       psym code+64+5            /e,f,m
```

```
/     finish
/     psym code+64
/     if text # 0
/       psym au
/       cycle
/         sym = b(text)
/         exit if sym = 0
/         psym sym
/         text = text-1
/       repeat
/       psym au
/     finish
/     newline
/     selin 0
/     ignore
xer:
   temp = #erprom-#prom
   jump ernew


edfin: /Close files
   rsym                              /nl
   if topcb >= dcb0
     switch if sin # 0
     while lend # fend
       move
     repeat
     cycle                           /until => edstop
       puttop
     repeat
   finish
edstop:
   resetcursor; clearline
   stop

x0: /Base label for execution switch

xopen: /Open bracket
   n(text) = num
xquery:
   jump nxt

xclose: /Close bracket
   attend; jump new if 3
   num = num-1
   jump xquery if num = 0
   n(ci-6) = num

xcomma: /+Comma
   ci = text
   jump xquery

xinv: /Invert
   jump no

xc: /Case change with right shift
   jump no if fp = lend
   sym = b(fp)&95
   if 'A' <= sym and sym <= 'Z'
     min = pp if pp < min
     sym = b(fp)\32
     jumps xr1
   finish

xr: /Right shift
```

```
    sym = b(fp)\32
    jumps xr1
  finish

xr: /Right shift
  jump no if fp = lend
  sym = b(fp)
xr1:
  b(pp) = sym
  pp = pp+1; fp = fp+1
  jump ok

xl: /Left shift
  jump no if pp = lbeg
  fp = fp-1; pp = pp-1
  b(fp) = b(pp)
  ms = 0
  jump ok

xe: /Erase
  jump no if fp = lend
  fp = fp+1
  jump oka

xeb: /Erase back
  jump no if pp = lbeg
  pp = pp-1
  jump oka

xs: /Substitute
  jump no if fp # ms
  fp = fp+ml
  min = pp if pp < min

xi: /+Insert
  jump no if pp-lbeg+lend-fp > maxlintmaxlin
  cycle
    sym = b(text)
    exit if sym = 0
    insert
    text = text-1
  repeat
  text = t(ci-6)
  jump ok

xw: xo: /Write, Overwrite
  prompt #anyprom
  cycle
    update
    zdisplay
    rsym
    jump nxt if sym < ' '
    if sym # del
      insert
      fp = fp+1 if code = 'o'&31 and fp # lend
    else if pp # lbeg
      pp = pp-1; min = pp
      if code = 'o'&31
        fp = fp-1; b(fp) = b(pp)
      finish
    finish
  repeat

xg: /Get
  lstar
  sym = nl; insert              /Create blank line
  lend = fp-1
  lstar
  killed = killed-1
  update
  zdisplay
```

```
   prompt 0
   cycle
     rsym
     exit if sym < ' '
     insert
   repeat
   min = pp
   if b(lbeg) = ':'
     pp = lbeg
     kill
     jump no
   finish
   min = inf
   move
   jump ok

xb: /Break
   sym = nl
   if num <= 0
     sym = ff; num = 1
   finish
   insert
   killed = killed-1
   update
   lbeg = pp
   line = line+1
   jump ok

xj: /Join
   rstar
   jump no if pp-lbeg > maxlin
   jumps xk1
 xk: /Kill
   pp = lbeg; fp = lend
xk1:
   kill
   jump ok

 xp: /Print
   update
   ac = num-1; vline = 0
   pdisplay
   prompt #prom
   match nl
   jump nxt if so
   jump new

 xm: /Move
   move
   jump ok

 xmb: /Move back
   jump no if sin # 0
   moveback
   jump ok

 xv:  /Verify
   verify
   jump no if not
   jump ok

xd: xt:                          /Delete, Traverse
   limit = c(ci-6)>>5
   fp1 = fp
   jump xf2
```

```
xu: xt:                          /Delete, Traverse
    limit = c(ci-6)>>5
    fp1 = fp
    jump xf2

 xfb: /Find back
    jump no if sin # 0
 xf: xu:                         /+Find, Uncover
    limit = c(ci-6)>>5
xf1:fp1 = fp
    jumps xf3 if fp = ms
 xf2:fp = fp-1
xf3:ac = b(text)
 xf4:fp = fp+1
    jumps xf5 if fp = lend
    jump xf4 if b(fp)\ac&(\32) # 0
    verify
    jump xf3 if not
    jump xf9
 xf5:fp = fp1
    limit = limit-1
    jump no if limit = 0
    if code = 'A'&31             /Find back
      moveback
    else if code = 'U'&31
      kill
    else
      move
    finish
    jump xf1
xf9:
    jump oka if code > 'T'&31    /Uncover
    fp = fptp if code = 'I'&31
    work = fp1
    while work # fp
      b(pp) = b(work)
      pp = pp+1; work = work+1
    repeat
    jump ok if code # 'D'&31
    fp = fptp
    jump oka

/Execution switch vector
 xsw:b #xeb-#x0>>1, #xfb-#x0>>1, #xb-#x0>>1, #xc-#x0>>1,
    #xd-#x0>>1, #xe-#x0>>1, #xf-#x0>>1, #xg-#x0>>1,
    #xmb-#x0>>1, #xi-#x0>>1, #xj-#x0>>1, #xk-#x0>>1,
    #xl-#x0>>1, #xm-#x0>>1, #xw-#x0>>1, #xo-#x0>>1,
    #xp-#x0>>1, #x0-#x0>>1, #xr-#x0>>1, #xs-#x0>>1,
    #xt-#x0>>1, #xu-#x0>>1, #xv-#x0>>1, #xw-#x0>>1,
    #x0-#x0>>1, #x0-#x0>>1, #x0-#x0>>1, #xopen-#x0>>1,
    #xinv-#x0>>1, #xclose-#x0>>1, #xcomma-#x0>>1, #xquery-#x0>>1

 $end editor

$end resident utilities

 $redef ltemp=*1
$list *1&(\1)
 <= 2C00
 b  reslim-* $ 0

 $list ltemp

$begin system initialisation
 $def ac=r0, temp=r1
 $def work=r4, work2=r5, cb=r6, tcb=r7
$def bit=r8, free=r9, pros=r10, dcbs=r11
 $def lr=r12, lt=r13, blk=r14,top=r14
 $temp r1
$def newpb=r13                   /*nb*
   lm r14,w(x'34')               /preserve 34,36
```