```
   let NormalSend[x, y] be
   §NS
       Symb := x
 5     Out[Output, x ∨ y]
   $NS


   and SendNL[x, y] be
10 §SN
       if x = GET do
       §   SaveLN := 1
           return
       $
15     if x = NEWLINE do
       §   SaveLN := y
           return
       $
       Send := NormalSend
20     if CanStartCom[x] do Send[SEMICOLON, 0]
       Send[NEWLINE, OldLN]
       unless GetNo = OldGet do
       §   Send[GET, OldGet]
           Send[GET, GetNo]
25     $
       Send[NEWLINE, SaveLN]
       Send[x, y]
   $SN

30
   and NLRt[] be
   §N
       OldLN := LineNo
       §   LineNo := LineNo + 1
35         NextCh[]
       $ repeatwhile Ch = '*n'
       if LineNo > 2000 do Error[3, HARD, DUMMY, TermPP]
       test CanEndCom[Symb]
         ifso  Send, SaveLN, Symb, OldGet := SendNL, LineNo, 0, GetNo
40               || Symb must not be able to end an exp or com
         ifnot §   Send[NEWLINE, OldLN]
                   Send[NEWLINE, LineNo]
   $N

45
   let GetRt[] be
   §G
       let i, g = 0, GetList↓0
       let v, End = GetList + g × GETSIZE, GetList + GETLISTSIZE
50     OldLN := LineNo
       if g ≥ GETMAX do Error[65, HARD, DUMMY, TermPP]
       §R
           Delete[CAT_b]
           if Ch = '|' do
55         §   NextCh[]
               if Wrong['|'] do
               §   Error[63, HARD, DUMMY, Delete, CAT_c]
                   return
               $
60             Delete[CAT_c]
           $
```

```
            if Ch = '*n' break
            unless Is[Ch, CAT_s] do
            §   Error[66, VERYHARD, DUMMY, Delete, CAT_c]
65              return
            §
            if v + i ≥ End do
            §   Error[67, VERYHARD, DUMMY, Delete, CAT_c]
                return
70          §
            i := i + 1
            v↓i := TreatString[Ch = '[' → ']', Ch]
        §R repeat
        if i = 0 do
75      §   Error[68, VERYHARD, DUMMY, NullProgram]
            return
        §
        unless IsRoomforInputStream[] do
        §   Error[69, VERYHARD, DUMMY, NullProgram]
80          return
        §
        test i = 1
          ifso  i, v↓2 := 2, 'Text'
          ifnot if (i ∧ 1) ≠ 0 do
85              §   i := i + 1
                    v↓i := 'Index'
                §
      § let File = FindFile[v, i]
        if File = 0 return
90      v↓FILENO, v↓SAVEIN, v↓SAVELINENO, v↓GETPRE :=
            File, In, LineNo, GetNo
        In := BytesfromWords[InfromFile[File]]
        if Endof[In] do
        §   Error[70, SOFT, DUMMY, NullProgram]
95          Close[In]
            In := v↓SAVEIN
            Ch := '*n'
            return
        §
100     Send[NEWLINE, OldLN]
        Send[GET, GetNo]
        GetList↓0 := g + 1
        GetNo := g + 1
        ProcVec↓ENDCH := DenestGets
105     NextCh[] repeatwhile Ch = '*n'
        LineNo := 1
        Send[GET, GetNo]
        Send[NEWLINE, 1]
    §G
110


    and FindFile[v, i] = valof
    §FF
        let Index = -1
115     while i > 2 do
        §   Index := SpecialLookUp[v↓(i - 1), v↓i, Index, true]
            i := i - 2
            if Index = 0 do resultis 0
        §
120   § let f = SpecialLookUp[v↓1, v↓2, Index, false]
        if f = 0 do resultis 0
      § let h = FindHeadingVector[f]
        if Deleted[h] do
        §   Error[76, VERYHARD, DUMMY, NullProgram]
125         f := 0
        §
```

```
        if Archived[h] do
        §   Error[77, VERYHARD, DUMMY, NullProgram]
            f := 0
130     §
        ReturnHeadVec[h]
        resultis f
    $FF

135
    and SpecialLookUp[S1, S2, i, IndReq] = valof
    §SLU
        let f = 0
        test i = -1
140       ifso  §   f := LookUp[S1, S2, CurrentIndex]
                    if f = 0 do
                        f := LookUp[S1, S2, IndReq → SystemIndex, DeclIndex]
              §
          ifnot f := LookUp[S1, S2, i]
145     if f = 0 do Error[IndReq → 71, 72, VERYHARD, DUMMY, NullProgram]
        resultis f
    $SLU


150 and DenestGets[] be
    §   let g = GetList + (GetNo - 1) × GETSIZE
        Close[In]
        Send[NEWLINE, LineNo]
        Send[GET, GetNo]
155     In, LineNo := g↓SAVEIN, g↓SAVELINENO
        GetNo := g↓GETPRE
        if GetNo = 0 do ProcVec↓ENDCH := P_End_of_Input
        Ch := '*n'                  || so next char sent gives correct line no
        Send[GET, GetNo]
160 §


    and CharacterRt[] be
    §CR
165     let RememberLN, SaveStack = LineNo, Stack
        let s = TreatString[Ch]
        if s = 0 return
        unless (s↓0 rshift 8) ≤ 1 do
        §   Error[73, HARD, DUMMY, NullProgram]
170         Send[ERROR, 0]
            return
        §
        Send[CHARACTER, s↓0 ∧ 8377]
        unless RememberLN = LineNo ∨ Ch = '*n' do Write[NEWLINE ∨ LineNo]
175                                     || Send without setting Symb
        Stack := SaveStack
    §CR


180 and StringRt[] be
    §SR
        let RememberLN = LineNo
        let s = TreatString[Ch = '[' → ']', Ch]
        if s = 0 do
185     §   Send[ERROR, 0]
            return
        §
        Push[s]
        Send[STRING, Adjoin_to[StringTable]]
190     unless RememberLN = LineNo ∨ Ch = '*n' do Write[NEWLINE ∨ LineNo]
                                        || Send without setting Symb
```

```
    $SR


195 and TreatString[EndQ] = valof
    §TS
        NextCh[]
        switchon Ch into
        §SW
200         case `**`:
                    Push[StarFn[Next[In]]]
                    endcase

            case `*n`:
205                 LineNo := LineNo + 1
                    NextCh[] repeatwhile Is[Ch, CAT_b]
                    endcase

            case ENDCH:
210                 if Endof[In] do
                    §   Error[59, VERYHARD, DUMMY, NullProgram]
                        ResetStack[]
                        resultis 0
                    §
215
            default:if Ch = EndQ break
                    PushCh_NextCh[]
        $SW repeat
        if TooBig[] do Error[58, VERYHARD, DUMMY, NullProgram]
220     NextCh[]
        resultis PackedString[]
    $TS


225 and StarFn[x] = valof
    §SF
        test x = `8`
          ifso  §8
                    let n = 0
230                 NextCh[] repeatwhile Is[Ch, CAT_b]
                    unless Is[Ch, CAT_o] do
                    §   Error[74, HARD, DUMMY, NullProgram]
                        resultis Ch
                    §
235                 for i = 1 to 3 do
                    §   n := (n lshift 3) + Ch - `0`
                        NextCh[]
                        unless Is[Ch, CAT_o] break
                    §
240                 resultis n
                $8
          ifnot §   NextCh[]
                    switchon x into
                    §S
245                     case `4`:
                                resultis `*4`

                        case `b`:
                                resultis `*b`
250
                        case `n`:
                                resultis `*n`

                        case `p`:
255                             resultis `*p`
```

```
                        case ‘q‘:
                                resultis ‘¿

260                     case ‘s‘:
                                resultis ‘*s‘

                        case ‘**‘:
                        case ‘*‘‘:
265                     case ‘*’‘:
                                resultis x

                        default:Error[57, HARD, x, NullProgram]
                                resultis x
270 $SF


    and PackedString[] = valof
    §PS
275     let s = Stack
        if TooBig[] do StackPtr := MAXLENGTH
        PackStack[]
        StepOnStack[]
        resultis s
280 $PS


    and NumFn[Base, Cat] = valof
    §NF
285     let n = 0
        while Is[Ch, Cat] do
        §   n := n × Base + Ch - ‘0‘
            NextCh[]
        §
290     if 0 ≤ n ≤ 500 do
        §   Send[NUMBER, n]
            resultis false
        §
        Push[n]
295     resultis true
    $NF


    let SendTables[] be
300 §ST
        SendT[’Numbers’, NumTable, Write]
        SendT[’Strings’, StringTable, SendS]
        SendGets[]
        SendIds[]
305 $ST


    and SendT[WF, Table, Rt] be
    §ST
310     Output := OuttoFile[WorkFile[WF]]
        Write[Table↓PTR - OFFSET]
        for i = FIRSTEL to Table↓PTR do Rt[Table↓i]
        Close[Output]
    $ST
315

    and SendS[s] be TransferOut[Output, s, (s↓0 rshift 9) + 1]


320 and SendGets[] be
    §SG
```

```
        Output := OuttoFile[WorkFile['Gets']]
        Write[GetList↓0]
        for i = 0 to GetList↓0 - 1 do
325     §   Write[GetList↓(i × GETSIZE + FILENO)]
            SendS[GetList↓(i × GETSIZE + 1)]
        §
        Close[Output]
    $SG
330

    and SendIds[] be
    §SI
        Output := OuttoFile[WorkFile['Names']]
335     Write[NameTable↓PTR - OFFSET + 26]
        for i = 0 to 25 do Write[(8400 ∨ 'a') + i]
        for i = FIRSTEL to NameTable↓PTR do SendS[NameTable↓i]
        Close[Output]
    $SI
340
****
```