```
program X1_ALGOL_60_compiler(input,output,lib_tape);

const d2  =         4;
      d3  =         8;
      d4  =        16;
      d5  =        32;
      d6  =        64;
      d7  =       128;
      d8  =       256;
      d10 =      1024;
      d12 =      4096;
      d13 =      8192;
      d15 =     32768;
      d16 =     65536;
      d17 =    131072;
      d18 =    262144;
      d19 =    524288;
      d20 =   1048576;
      d21 =   2097152;
      d22 =   4194304;
      d23 =   8388608;
      d24 =  16777216;
      d25 =  33554432;
      d26 =  67108864;
      mz  = 134217727;

      gvc0 =      138;   {0-04-10}
      tlib =      800;   {0-25-00}
      plie =     6783;   {6-19-31}
      bim  =      930;   {0-29-02}
      nlscop =     31;
      nlsc0 =      48;
      mlib =      800;   {0-25-00}
      klie =    10165;   {9-29-21}
      crfb =      623;   {0-19-15}
      mcpb =      928;   {0-29-00}

var tlsc,plib,flib,klib,nlib,
    rht,vht,qc,scan,rfsb,rnsa,rnsb,rnsc,rnsd,
    dl,inw,fnw,dflag,bflag,oflag,
    nflag,kflag,
    iflag,mflag,vflag,aflag,sflag,eflag,jflag,pflag,fflag,
    bn,vlam,pnlv,gvc,lvc,oh,id,nid,ibd,
    inba,fora,forc,psta,pstb,spe,
    arra,arrb,arrc,arrd,ic,aic,rlaa,rlab,qa,qb,
    rlsc,flsc,klsc,nlsc: integer;
    bitcount,bitstock: integer;
    store: array[0..12287] of integer;
    rns_state: (ps,ms,virginal);
    rfs_case,nas_stock,pos: integer;
    word_del_table: array[10..38] of integer;
    flex_table: array[0..127] of integer;
    opc_table: array[0..112] of integer;

    rlib,mcpe: integer;

    lib_tape: text;

    ii: integer;

procedure stop(n: integer);
{emulation of a machine instruction}
begin writeln(output);
  writeln(output,'*** stop ',n div d5:1,'-',n mod d5:2,' ***');
  halt
end {stop};

function read_flexowriter_symbol: integer;                          {LK}
label 1,2;
var s,fts: integer;
begin
  1: read(input,s);
     if rfsb = 0
     then if (s = 62 {tab}) or (s = 16 {space}) or (s = 26 {crlf})
          then goto 2
          else if (s = 122 {lc}) or (s = 124 {uc}) or (s = 0 {blank})
               then begin rfsb:= s {new flexowriter shift}; goto 1 end
               else if s = 127 {erase} then goto 1
               else stop(19) {flexowriter shift undefined};
  2: fts:= flex_table[s];
     if fts > 0
     then if rfsb = 124
          then {uppercase} read_flexowriter_symbol:= fts div d8
          else {lowercase} read_flexowriter_symbol:= fts mod d8
     else if fts = -0 then stop(20) {wrong parity}
     else if fts = -1 then stop(21) {undefined punching}
     else if s = 127 {erase} then goto 1
     else begin rfsb:= s {new flexowriter shift}; goto 1 end
end {read_flexowriter_symbol};
```

```
function next_ALGOL_symbol: integer;                                 {HT}
label 1;
var sym,wdt1,wdt2: integer;
begin sym:= - nas_stock;
  if sym >= 0 {symbol in stock}
  then nas_stock:= sym + 1{stock empty now}
  else sym:= read_flexowriter_symbol;
1: if sym > 101 {analysis required}
  then begin if sym = 123 {space symbol} then sym:= 93;
         if sym <= 119 {space symbol, tab, or nlcr}
         then if qc = 0
              then begin sym:= read_flexowriter_symbol;
                     goto 1
                   end
              else
         else if sym = 124 {:}
              then begin sym:= read_flexowriter_symbol;
                     if sym = 72
                     then sym:= 92 {:=}
                     else begin nas_stock:= -sym; sym:= 90 {:} end
                   end
         else if sym = 162 {|}
              then begin repeat sym:= read_flexowriter_symbol
                     until sym <> 162;
                     if sym = 77 {^} then sym:= 69 {|^}
                     else if sym = 72 {=} then sym:= 75 {|=}
                     else if sym = 74 {<} then sym:= 102 {|<}
                     else if sym = 70 {>} then sym:= 103 {|>}
                     else stop(11)
                   end
         else if sym = 163 {_}
           then begin repeat sym:= read_flexowriter_symbol
                  until sym <> 163;
                  if (sym > 9) and (sym <= 38) {a..B}
                  then begin {word delimiter}
                         wdt1:= word_del_table[sym] mod 128;
                         if wdt1 >= 63
                         then sym:= wdt1
                         else if wdt1 = 0
                         then stop(13)
                         else if wdt1 = 1 {sym = c}
                         then if qc = 0 {outside string}
                           then begin {skip comment}
                                  repeat sym:= read_flexowriter_symbol
                                  until sym = 91 {;};
```

```
                sym:= read_flexowriter_symbol;
                goto 1
              end
         else sym:= 97 {comment}
       else begin sym:= read_flexowriter_symbol;
              if sym = 163 {_}
              then begin repeat sym:=
                       read_flexowriter_symbol
                     until sym <> 163;
                     if (sym > 9) and (sym <= 32)
                     then if sym = 29 {t}
                       then begin sym:=
                                read_flexowriter_symbol;
                              if sym = 163 {_}
                              then begin repeat
                                       sym:=
                                       read_flexowriter_symbol
                                     until sym <> 163;
                                     if sym = 14 {e}
                                     then sym:=  94 {step}
                                     else sym:= 113 {string}
                                   end
                              else stop(12)
                            end
                       else begin wdt2:=
                                word_del_table[sym] div 128;
                              if wdt2 = 0
                              then sym:= wdt1 + 64
                              else sym:= wdt2
                            end
                     else stop(13)
                   end
              else stop(12)
            end;
       repeat nas_stock:=  - read_flexowriter_symbol;
         if nas_stock = - 163 {_}
         then repeat nas_stock:= read_flexowriter_symbol
           until nas_stock <> 163
       until nas_stock <= 0
     end {word delimiter}
else if sym = 70 {>} then sym:= 71 {>=}
else if sym = 72 {=} then sym:= 80 {eqv}
else if sym = 74 {<} then sym:= 73 {<=}
else if sym = 76 {~} then sym:= 79 {imp}
else if sym = 124 {:} then sym:= 68 {div}
```

```
                  else stop(13)
                end
         else stop(14) {? or " or '}
       end;
  next_ALGOL_symbol:= sym
end {next_ALGOL_symbol};

procedure read_next_symbol;                                         {ZY}
label 1;
begin
1: case rns_state of
  ps: begin dl:= next_ALGOL_symbol;
        {store symbol in symbol store:}
        if rnsa > d7
        then begin rnsa:= rnsa div d7;
               store[rnsb]:= store[rnsb] + dl * rnsa
             end
        else begin rnsa:= d15; rnsb:= rnsb + 1; store[rnsb]:= dl * rnsa;
               if rnsb + 8 > plib then stop(25)
             end
      end;
  ms: begin {take symbol from symbol store:}
        dl:= (store[rnsd] div rnsc) mod d7;
        if rnsc > d7
        then rnsc:= rnsc div d7
        else begin rnsc:= d15; rnsd:= rnsd + 1 end
      end;
  virginal:
      begin qc:= 0; rfs_case:= 0; nas_stock:= 1;
        if scan > 0 {prescan}
        then begin rns_state:= ps;
               {initialize symbol store:}
               rnsb:= bim + 8; rnsd:= bim + 8; rnsa:= d22; rnsc:= d15;
               store[rnsb]:= 0;
             end
        else rns_state:= ms;
        goto 1
      end
  end {case}
end {read_next_symbol};

procedure read_until_next_delimiter;                                {FT}
  label 1,3,4,5;
  var marker,elsc,bexp: integer;
  function test1: boolean;
  begin if dl = 88 {.}
    then begin dflag:= 1;
           read_next_symbol; test1:= test1
         end
    else if dl = 89 {ten} then goto 1
    else test1:= dl > 9
  end {test1};

  function test2: boolean;
  begin if dl = 89 {ten} then inw:= 1; test2:= test1
  end {test2};

  function test3: boolean;
  begin read_next_symbol; test3:= test1
  end {test3};

begin {body of read_until_next_delimiter}
  read_next_symbol;
  nflag:= 1;
  if (dl > 9) and (dl < 63) {letter}
  then begin dflag:= 0; kflag:= 0; inw:= 0;
         repeat fnw:= (inw mod d6) * d21; inw:= inw div d6 + dl * d21;
           read_next_symbol
         until (inw mod d3 > 0) or (dl > 62);
         if inw mod d3 > 0
         then begin dflag:= 1;
                fnw:= fnw + d23; marker:= 0;
                while (marker = 0) and (dl < 63) do
                begin marker:= fnw mod d6 * d21; fnw:= fnw div 64 + dl * d21;
                  read_next_symbol
                end;
                while marker = 0 do
                begin marker:= fnw mod d6 * d21;
                  fnw:= fnw div d6 + 63 * d21
                end;
                while dl < 62 do read_next_symbol
              end;
         goto 4;
       end;
  kflag:= 1; fnw:= 0; inw:= 0; dflag:= 0; elsc:= 0;
  if test2 {not (dl in [0..9,88,89])}
  then begin nflag:= 0;
         if (dl = 116 {true}) or (dl = 117 {false})
         then begin inw:= dl - 116;
```

```
                 dflag:= 0; kflag:= 1; nflag:= 1;
                 read_next_symbol;
                 goto 4
               end;
          goto 5
        end;
    repeat if fnw < d22
      then begin inw:= 10 * inw + dl;
             fnw:= 10 * fnw + inw div d26;
             inw:= inw mod d26;
             elsc:= elsc - dflag
           end
      else elsc:= elsc - dflag + 1
    until test3;
    if (dflag = 0) and (fnw = 0)
    then goto 4;
    goto 3;
  1: if test3 {not (dl in [0..9,88,89]}
    then if dl = 64 {plus}
         then begin read_next_symbol; dflag:= dl end
         else begin read_next_symbol; dflag:= - dl - 1 end
    else dflag:= dl;
    while not test3 {dl in [0..9,88,89]} do
    begin if dflag >= 0
      then dflag:= 10 * dflag + dl
      else dflag:= 10 * dflag - dl + 9;
      if abs(dflag) >= d26 then stop(3)
    end;
    if dflag < 0 then dflag:= dflag + 1;
    elsc:= elsc + dflag;
  3: {float}
    if (inw = 0) and (fnw = 0)
    then begin dflag:= 0; goto 4 end;
    bexp:= 2100 {2**11 + 52; P9-characteristic};
    while fnw < d25 do
    begin inw:= 2 * inw; fnw:= 2 * fnw + inw div d26; inw:= inw mod d26;
      bexp:= bexp - 1
    end;
    if elsc > 0
    then repeat fnw:= 5 * fnw; inw:= (fnw mod 8) * d23 + (5 * inw) div 8;
           fnw:= fnw div 8;
           if fnw < d25
           then begin inw:= 2 * inw; fnw:= 2 * fnw + inw div d26;
                  inw:= inw mod d26;
                  bexp:= bexp - 1
```

```
              end;
         bexp:= bexp + 4; elsc:= elsc - 1;
       until elsc = 0
  else if elsc < 0
  then repeat if fnw >= 5 * d23
         then begin inw:= inw div 2 + (fnw mod 2) * d25;
                fnw:= fnw div 2; bexp:= bexp + 1
              end;
         inw:= 8 * inw; fnw:= 8 * fnw + inw div d26;
         inw:= inw mod d26 + fnw mod 5 * d26;
         fnw:= fnw div 5; inw:= inw div 5;
         bexp:= bexp - 4; elsc:= elsc + 1
       until elsc = 0;
  inw:= inw + 2048;
  if inw >= d26
  then begin inw:= 0; fnw:= fnw + 1;
         if fnw = d26 then begin fnw:= d25; bexp:= bexp + 1 end
       end;
  if (bexp < 0) or (bexp > 4095) then stop(4);
  inw:= (inw div 4096) * 4096 + bexp;
  dflag:= 1;
4: oflag:= 0;
5:
end {read_until_next_delimiter};

procedure fill_t_list(n: integer);
begin store[tlsc]:= n; tlsc:= tlsc + 1
end {fill_t_list};


procedure prescan;                                                  {HK}

  label 1,2,3,4,5,6,7;
  var bc,mbc: integer;

  procedure fill_prescan_list(n: integer); {n = 0 or n = 1}         {HF}
    var i,j,k: integer;
  begin {update plib and prescan_list chain:}
    k:= plib; plib:= k - dflag - 1; j:= k;
    for i:= 2*bc + n downto 1 do
    begin k:= store[j]; store[j]:= k - dflag - 1; j:= k end;
    {shift lower part of prescan_list down over dfag + 1 places:}
    k:= plib;
    if dflag = 0
    then for i:= j - plib downto 1 do
```

```
         begin store[k]:= store[k+1]; k:= k + 1 end
    else begin {shift:}
           for i:= j - plib - 1 downto 1 do
           begin store[k]:= store[k+2]; k:= k + 1 end;
           {enter fnw in prescan_list:}
           store[k+1]:= fnw
         end;
    {enter inw in prescan_list:}
    store[k]:= inw
  end {fill_prescan_list};

  procedure augment_prescan_list;                                  {HH}
  begin dflag:= 1; inw:= plie; fnw:= plie - 1;
    fill_prescan_list(0)
  end {augment_prescan_list};

  procedure block_introduction;                                    {HK}
  begin fill_t_list(bc); fill_t_list(-1) {block-begin marker};
    mbc:= mbc + 1; bc:= mbc;
    augment_prescan_list
  end {block_introduction};

begin {body of prescan}
  plib:= plie; store[plie]:= plie - 1; tlsc:= tlib;
  bc:= 0; mbc:= 0; qc:= 0; rht:= 0; vht:= 0;
  fill_t_list(dl); {dl should be 'begin'}
  augment_prescan_list;
1: bflag:= 0;
2: read_until_next_delimiter;
3: if dl <= 84 {+,-,*,/,_:,|^,>,>=,=,<=,<,|=,~,^,`,_~,_=,goto,if,then,else}
  then {skip:} goto 1;
  if dl = 85 {for}
  then begin block_introduction; goto 1 end;
  if dl <= 89 {do,comma,period,ten} then {skip:} goto 1;
  if dl = 90 {:} then begin fill_prescan_list(0); goto 2 end;
  if dl = 91 {;}
  then begin while store[tlsc-1] < 0 {block-begin marker} do
             begin tlsc:= tlsc - 2; bc:= store[tlsc] end;
         if rht <> 0 then stop(22); if vht <> 0 then stop(23);
         goto 1
       end;
  if dl <= 97 {:=,step,until,while,comment} then {skip:} goto 1;
  if dl <= 99 {(,)}
  then begin if dl = 98 then rht:= rht + 1 else rht:= rht - 1;
         goto 1
       end;
  if dl <= 101 {[,]}
  then begin if dl = 100 then vht:= vht + 1 else vht:= vht - 1;
         goto 1
       end;
  if dl = 102 {|<}
  then begin repeat if dl = 102 {|<} then qc:= qc + 1;
               if dl = 103 {|>} then qc:= qc - 1;
               if qc > 0 then read_next_symbol
             until qc = 0;
         goto 2
       end;
  if dl = 104 {begin}
  then begin fill_t_list(dl);
         if bflag <> 0 then goto 1;
         read_until_next_delimiter;
         if (dl <= 105) or (dl > 112) then goto 3;
         tlsc:= tlsc - 1 {remove begin from t_list};
         block_introduction;
         fill_t_list(104) {add begin to t_list again};
         goto 3;
       end;
  if dl = 105 {end}
  then begin while store[tlsc-1] < 0 {block-begin marker} do
             begin tlsc:= tlsc - 2; bc:= store[tlsc] end;
         if rht <> 0 then stop(22); if vht <> 0 then stop(23);
         tlsc:= tlsc - 1 {remove corresponding begin from t_list};
         if tlsc > tlib then goto 1;
         goto 7 {end of prescan}
       end;
  if dl <= 105 {dl = |>} then goto 1;
  if dl = 111 {switch}
  then if bflag = 0
       then {declarator}
            begin read_until_next_delimiter {for switch identifier};
              fill_prescan_list(0); goto 6
            end
       else {specifier}
            goto 5;
4: if dl = 112 {procedure}
   then if bflag = 0
        then {declarator}
             begin bflag:= 1;
               read_until_next_delimiter {for procedure identifier};
               fill_prescan_list(1); block_introduction; goto 6
            end
         else {specificier}
            goto 5;
   if dl > 117 {false} then stop(8);
5: read_until_next_delimiter;
6: if dl <> 91 {;} then goto 4;
   goto 2;
7:
end {prescan};


procedure intro_new_block2;                                      {HW}
label 1;
var i,w: integer;
begin inba:= d17 + d15;
1: i:= plib; plib:= store[i]; i:= i + 1;
  while i <> plib do
  begin w:= store[i];
    if w mod 8 = 0 {at most 4 letters/digits}
    then i:= i + 1
    else begin store[nlib+nlsc]:=store[i+1]; i:= i + 2; nlsc:= nlsc + 1 end;
    store[nlib+nlsc]:= w; nlsc:= nlsc + 2;
    if nlib + nlsc > i then stop(15);
    store[nlib+nlsc-1]:= bn * d19 + inba
  end;
  if inba <> d18 + d15
  then begin inba:= d18 + d15; goto 1 end;
  lvc:= 0
end {intro_new_block2};

procedure intro_new_block1;                                      {HW}
begin fill_t_list(nlsc); fill_t_list(161);
  intro_new_block2
end {intro_new_block1};

procedure intro_new_block;                                       {HW}
begin bn:= bn + 1; intro_new_block1
end {intro_new_block};

procedure bit_string_maker(w: integer);                          {LL}
var head,tail,i: integer;
begin head:= 0; tail:= w mod d10;
  {shift (head,tail) bitcount places to the left:}
  for i:= 1 to bitcount do
  begin head:= 2 * head + tail div d26; tail:= (tail mod d26) * 2
  end {shift};
  bitstock:= bitstock + tail; bitcount:= bitcount + w div d10;
  if bitcount > 27
  then begin bitcount:= bitcount - 27;
         store[rnsb]:= bitstock; bitstock:= head; rnsb:= rnsb + 1;
         if rnsb = rnsd
         then if nlib + nlsc + 8 < plib
              then begin {shift text, fli, kli and nli}
                     for i:= nlib + nlsc - rnsd - 1 downto 0 do
                     store[rnsd+i+8]:= store[rnsd+i];
                     rnsd:= rnsd + 8; flib:= flib + 8;
                     klib:= klib + 8; nlib:= nlib + 8
                   end
              else stop(25)
       end
end {bit_string_maker};

procedure address_coder(a: integer);                                {LS}
var w: integer;
begin w:= a mod d5;
  if w = 1 then w:= 2048 {2*1024 +  0} else
  if w = 2 then w:= 3074 {3*1024 +  2} else
  if w = 3 then w:= 3075 {3*1024 +  3}
           else w:= 6176 {6*1024 + 32} + w;
  bit_string_maker(w);
  w:= (a div d5) mod d5;
  if w = 0 then w:= 2048 {2*1024 +  0} else
  if w = 1 then w:= 4100 {4*1024 +  4} else
  if w = 2 then w:= 4101 {4*1024 +  5} else
  if w = 4 then w:= 4102 {4*1024 +  6} else
  if w = 5 then w:= 4103 {4*1024 +  7}
           else w:= 6176 {6*1024 + 32} + w;
  bit_string_maker(w);
  w:= (a div d10) mod d5;
  if w = 0 then w:= 1024 {1*1024 + 0}
           else w:= 6176 {6*1024 + 32} + w;
  bit_string_maker(w)
end {address_coder};

procedure fill_result_list(opc,w: integer);                         {ZF}
var j: 8..61;
begin rlsc:= rlsc + 1;
  if opc < 8
  then begin address_coder(w);
         w:= (w div d15) * d15 + opc;
       if w = 21495808 {  2S   0 A  } then w:= 3076 {3*1024 +   4} else
       if w = 71827459 {  2B   3 A  } then w:= 3077 {3*1024 +   5} else
       if w = 88080386 {  2T 2X0    } then w:= 4108 {4*1024 +  12} else
       if w = 71827456 {  2B   0 A  } then w:= 4109 {4*1024 +  13} else
       if w =  4718592 {  2A   0 A  } then w:= 7280 {7*1024 + 112} else
       if w = 71303170 {  2B 2X0    } then w:= 7281 {7*1024 + 113} else
       if w = 88604673 {  2T   1 A  } then w:= 7282 {7*1024 + 114} else
       if w =        0 {  0A 0X0    } then w:= 7283 {7*1024 + 115} else
       if w =   524291 {  0A   3 A  } then w:= 7284 {7*1024 + 116} else
       if w = 88178690 {N 2T 2X0    } then w:= 7285 {7*1024 + 117} else
       if w = 71827457 {  2B   1 A  } then w:= 7286 {7*1024 + 118} else
       if w =  1048577 {  0A 1X0 B  } then w:= 7287 {7*1024 + 119} else
       if w = 20971522 {  2S 2X0    } then w:= 7288 {7*1024 + 120} else
       if w =  4784128 {Y 2A   0 A  } then w:= 7289 {7*1024 + 121} else
       if w =  8388608 {  4A 0X0    } then w:= 7290 {7*1024 + 122} else
       if w =  4390912 {Y 2A 0X0   P} then w:= 7291 {7*1024 + 123} else
       if w = 13172736 {Y 6A   0 A  } then w:= 7292 {7*1024 + 124} else
       if w =  1572865 {  0A 1X0 C  } then w:= 7293 {7*1024 + 125} else
       if w =   524288 {  0A   0 A  } then w:= 7294 {7*1024 + 126}
       else begin address_coder(w div d15 + opc * d12);
              w:= 7295 {7*1024 + 127}
            end
     end {opc < 8}
else if opc <= 61
then begin j:= opc;
       case j of
          8: w:= 10624 {10*1024+384};  9: w:=  6160 { 6*1024+ 16};
         10: w:= 10625 {10*1024+385}; 11: w:= 10626 {10*1024+386};
         12: w:= 10627 {10*1024+387}; 13: w:=  7208 { 7*1024+ 40};
         14: w:=  6161 { 6*1024+ 17}; 15: w:= 10628 {10*1024+388};
         16: w:=  5124 { 5*1024+  4}; 17: w:=  7209 { 7*1024+ 41};
         18: w:=  6162 { 6*1024+ 18}; 19: w:=  7210 { 7*1024+ 42};
         20: w:=  7211 { 7*1024+ 43}; 21: w:= 10629 {10*1024+389};
         22: w:= 10630 {10*1024+390}; 23: w:= 10631 {10*1024+391};
         24: w:= 10632 {10*1024+392}; 25: w:= 10633 {10*1024+393};
         26: w:= 10634 {10*1024+394}; 27: w:= 10635 {10*1024+395};
         28: w:= 10636 {10*1024+396}; 29: w:= 10637 {10*1024+397};
         30: w:=  6163 { 6*1024+ 19}; 31: w:=  7212 { 7*1024+ 44};
         32: w:= 10638 {10*1024+398}; 33: w:=  4096 { 4*1024+  0};
         34: w:=  4097 { 4*1024+  1}; 35: w:=  7213 { 7*1024+ 45};
         36: w:= 10639 {10*1024+399}; 37: w:= 10640 {10*1024+400};
         38: w:= 10641 {10*1024+401}; 39: w:=  7214 { 7*1024+ 46};
         40: w:= 10642 {10*1024+402}; 41: w:= 10643 {10*1024+403};
         42: w:= 10644 {10*1024+404}; 43: w:= 10645 {10*1024+405};
         44: w:= 10646 {10*1024+406}; 45: w:= 10647 {10*1024+407};
```

```
            46: w:= 10648 {10*1024+408}; 47: w:= 10649 {10*1024+409};
            48: w:= 10650 {10*1024+410}; 49: w:= 10651 {10*1024+411};
            50: w:= 10652 {10*1024+412}; 51: w:= 10653 {10*1024+413};
            52: w:= 10654 {10*1024+414}; 53: w:= 10655 {10*1024+415};
            54: w:= 10656 {10*1024+416}; 55: w:= 10657 {10*1024+417};
            56: w:=  5125 { 5*1024+  5}; 57: w:= 10658 {10*1024+418};
            58: w:=  5126 { 5*1024+  6}; 59: w:= 10659 {10*1024+419};
            60: w:= 10660 {10*1024+420}; 61: w:=  7215 { 7*1024+ 47}
          end {case}
        end {opc <= 61}
  else if opc = 85{ST}
  then w:=  5127 { 5*1024 +   7}
  else w:= 10599 {10*1024 + 359} + opc;
  bit_string_maker(w)
end {fill_result_list};


procedure main_scan;                                                 {EL}

  label 1,2,3,64,66,69,70,76,81,82,8201,8202,83,8301,84,8401,85,8501,
        86,8601,87,8701,8702,8703,8704,8705,
        90,91,92,94,95,96,98,9801,9802,9803,9804,99,100,101,
        102,104,105,1052,106,107,108,1081,1082,1083,
        109,110,1101,1102,1103,111,112,1121,1122,1123,1124;

  procedure fill_t_list_with_delimiter;                              {ZW}
  begin fill_t_list(d8*oh+dl)
  end {fill_t_list_with_delimiter};

  procedure fill_future_list(place,value: integer);                  {FU}
  var i: integer;
  begin if place >= klib
    then begin if nlib + nlsc + 16 >= plib then stop(6);
           for i:= nlib + nlsc - 1 downto klib do
           store[i+16]:= store[i];
           klib:= klib + 16; nlib:= nlib + 16
         end;
    store[place]:= value
  end {fill_future_list};

  procedure fill_constant_list(n: integer);                          {KU}
  var i: integer;
  begin if klib + klsc = nlib
    then begin if nlib + nlsc + 16 >= plib then stop(18);
           for i:= nlib + nlsc - 1 downto nlib do
         store[i+16]:= store[i];
         nlib:= nlib + 16
       end;
  if n >= 0
  then store[klib+klsc]:= n
  else {one’s complement representation} store[klib+klsc]:= mz + n;
  klsc:= klsc + 1
end {fill_constant_list};

procedure unload_t_list_element(var variable: integer);           {ZU}
begin tlsc:= tlsc - 1; variable:= store[tlsc]
end {unload_t_list_element};

procedure fill_output(c: integer);
begin pos:= pos + 1;
  if c < 10 then write(chr(c+ord(’0’)))
  else if c < 36 then write(chr(c-10+ord(’a’)))
  else if c < 64 then write(chr(c-37+ord(’A’)))
  else if c = 184 then write(’ ’)
  else if c = 138
       then begin write(’ ’:8 - (pos - 1) mod 8);
              pos:= pos + 8 - (pos - 1) mod 8
            end
  else begin writeln; pos:= 0 end
end {fill_output};

procedure offer_character_to_typewriter(c: integer);              {HS}
begin c:= c mod 64;
  if c < 63 then fill_output(c)
end {offer_character_to_typewriter};

procedure label_declaration;                                      {FY}
var id,id2,i,w: integer;
begin id:= store[nlib+nid];
  if (id div d15) mod 2 = 0
  then begin {preceding applied occurrences}
         fill_future_list(flib+id mod d15,rlsc)
       end
  else {first occurrence}
       store[nlib+nid]:= id - d15 + 1 * d24 + rlsc;
  id:= store[nlib+nid-1];
  if id mod d3 = 0
  then begin {at most 4 letters/digits}
         i:= 4; id:= id div d3;
         while (id mod d6) = 0{void} do
```

```
            begin i:= i - 1; id:= id div d6 end;
            repeat offer_character_to_typewriter(id);
              i:= i - 1; id:= id div d6
            until i = 0
          end
    else begin id2:= store[nlib+nid-2];
            id2:= id2 div d3 + (id2 mod d3) * d24;
            w:= (id2 mod d24) * d3 + id div d24;
            id:= (id mod d24) * d3 + id2 div d24;
            id2:= w;
            i:= 9;
            repeat offer_character_to_typewriter(id);
              i:= i - 1;
              w:= id2 div d6 + (id mod d6) * d21;
              id:= id div d6 + (id2 mod d6) * d21;
              id2:= w
            until i = 0
          end;
    fill_output(138{TAB});
    w:= rlsc;
    for i:= 1 to 3 do
    begin offer_character_to_typewriter(w div d10 div 10);
      offer_character_to_typewriter(w div d10 mod 10);
      w:= (w mod d10) * d5;
      if i < 3 then fill_output(184{SPACE})
    end;
    fill_output(139{NLCR})
  end {label_declaration};

  procedure test_first_occurrence;                                  {LF}
  begin id:= store[nlib+nid];
    if (id div d15) mod 2 = 1 {first occurrence}
    then begin id:= id - d15 - id mod d15 + 2 * d24 + flsc;
            if nid <= nlsc0 {MCP}
            then fill_future_list(flib+flsc,store[nlib+nid]);
            store[nlib+nid]:= id;
            flsc:= flsc + 1
          end
  end {test_first_occurrence};

  procedure new_block_by_declaration1;                              {HU}
  begin fill_result_list(0,71827456+bn) {2B 'bn' A};
    fill_result_list(89{SCC},0);
    pnlv:= 5 * 32 + bn; vlam:= pnlv
  end {new_block_by_declaration1};
```

```
procedure new_block_by_declaration;                                   {HU}
begin if store[tlsc-2] <> 161{block-begin marker}
  then begin tlsc:= tlsc - 1 {remove 'begin'};
         fill_result_list(0,4718592) {2A 0 A};
         fill_result_list(1,71827456+rlsc+3) {2B 'rlsc+3' A};
         fill_result_list(9{ETMP},0);
         fill_result_list(2,88080384+flsc) {2T 'flsc'};
         fill_t_list(flsc); flsc:= flsc + 1;
         intro_new_block;
         fill_t_list(104{begin});
         new_block_by_declaration1
       end
end {new_block_by_declaration};

procedure fill_name_list;                                             {HN}
begin nlsc:= nlsc + dflag + 2;
  if nlsc + nlib > plib then stop(16);
  store[nlib+nlsc-1]:= id; store[nlib+nlsc-2]:= inw;
  if inw mod d3 > 0 then store[nlib+nlsc-3]:= fnw
end {fill_name_list};

procedure reservation_of_local_variables;                             {KY}
begin if lvc > 0
  then begin fill_result_list(0,4718592+lvc) {2A 'lvc' A};
         fill_result_list(0,8388657) {4A 17X1};
         fill_result_list(0,8388658) {4A 18X1}
       end
end {reservation_of_local_variables};

procedure address_to_register;                                        {ZR}
begin if id div d15 mod 2 = 0 {static addressing}
  then if id div d24 mod d2 = 2 {future list}
       then fill_result_list(2,
              71303168+id mod d15{2B 'FLI-address'})
       else fill_result_list(id div d24 mod 4,
              71827456+id mod d15{2B 'static address' A})
  else fill_result_list(0,
              21495808+id mod d15{2S 'dynamic address' A})
end {address_to_register};

procedure generate_address;                                           {ZH}
var opc: integer;
begin address_to_register;
  if (id div d16) mod 2 = 1
  then {formal} fill_result_list(18{TFA},0)
  else begin opc:= 14{TRAD};
         if (id div d15) mod 2 = 0 then opc:= opc + 1{TRAS};
         if (id div d19) mod 2 = 1 then opc:= opc + 2{TIAD or TIAS};
         fill_result_list(opc,0)
       end
end {generate_address};

procedure reservation_of_arrays;                                  {KN}
begin if vlam <> 0
  then begin vlam:= 0;
         if store[tlsc-1] = 161{block-begin marker}
         then rlaa:= nlib + store[tlsc-2]
         else rlaa:= nlib + store[tlsc-3];
         rlab:= nlib + nlsc;
         while rlab <> rlaa do
         begin id:= store[rlab-1];
           if (id >= d26) and (id < d25 + d26)
           then begin {value array:}
                  address_to_register;
                  if (id div d19) mod 2 = 0
                  then fill_result_list(92{RVA},0)
                  else fill_result_list(93{IVA},0);
                  store[rlab-1]:= (id div d15) * d15 - d16 + pnlv;
                  pnlv:= pnlv + 8 * 32 {at most 5 indices}
                end;
           if store[rlab-2] mod d3 = 0
           then rlab:= rlab - 2 else rlab:= rlab - 3
         end;
         rlab:= nlib + nlsc;
         while rlab <> rlaa do
         begin if store[rlab-1] >= d26
           then begin id:= store[rlab-1] - d26;
                  if id < d25
                  then begin address_to_register;
                         fill_result_list(95{VAP},0)
                       end
                  else begin id:= id - d25;
                         address_to_register;
                         fill_result_list(94{LAP},0)
                       end
                end;
           if store[rlab-2] mod d3 = 0
           then rlab:= rlab - 2 else rlab:= rlab - 3
         end;
```

```
              if nflag <> 0
              then id:= store[nlib+nid]
            end
    end {reservation_of_arrays};

    procedure procedure_statement;                                        {LH}
    begin if eflag = 0 then reservation_of_arrays;
      if nid > nlscop
      then begin if fflag = 0 then test_first_occurrence;
              address_to_register
            end
      else begin fill_t_list(store[nlib+nid] mod d12);
              if dl = 98{(}
              then begin eflag:= 1; goto 9801 end
            end
    end {procedure_statement};

    procedure production_transmark;                                       {ZL}
    begin fill_result_list(9+2*fflag-eflag,0)
    end {production_transmark};

    procedure production_of_object_program(opht: integer);                {ZS}
    var operator,block_number: integer;
    begin oh:= opht;
      if nflag <> 0
      then begin nflag:= 0; aflag:= 0;
              if pflag = 0
              then if jflag = 0
                   then begin address_to_register;
                          if oh > (store[tlsc-1] div d8) mod 16
                          then operator:= 315{5*63}
                          else begin operator:= store[tlsc-1] mod d8;
                                 if (operator <= 63) or (operator > 67)
                                 then operator:= 315{5*63}
                                 else begin tlsc:= tlsc - 1;
                                        operator:= 5 * operator
                                      end
                               end;
                          if fflag = 0
                          then begin if id div d15 mod 2 = 0
                                 then operator:= operator + 1;
                                 if id div d19 mod 2 <> 0
                                 then operator:= operator + 2;
                                 fill_result_list(operator-284,0)
                               end
```

```
                        else fill_result_list(operator-280,0)
                      end
                 else if fflag = 0
                      then begin block_number:= id div d19 mod d5;
                             if block_number <> bn
                             then begin fill_result_list
                                          (0,71827456+block_number);
                                    fill_result_list(28{GTA},0)
                                  end;
                             test_first_occurrence;
                             if id div d24 mod 4 = 2
                             then fill_result_list(2,88080384+id mod d15)
                                  {2T 'address'}
                             else fill_result_list(1,88604672+id mod d15)
                                  {2T 'address' A}
                           end
                      else begin address_to_register;
                             fill_result_list(35{TFR},0)
                           end
            else begin procedure_statement;
                   if nid > nlscop
                   then begin fill_result_list(0,4718592{2A 0 A});
                          production_transmark
                        end
                 end
          end
     else if aflag <> 0
     then begin aflag:= 0; fill_result_list(58{TAR},0) end;
     while oh <= store[tlsc-1] div d8 mod 16 do
     begin tlsc:= tlsc - 1; operator:= store[tlsc] mod d8;
       if (operator > 63) and (operator<= 80)
       then fill_result_list(operator-5,0)
       else if operator = 132 {NEG}
       then fill_result_list(57{NEG},0)
       else if (operator < 132) and (operator > 127)
       then begin {ST,STA,STP,STAP}
              if operator > 129
              then begin {STP,STAP}
                     tlsc:= tlsc - 1;
                     fill_result_list(0,71827456+store[tlsc]{2B 'BN' A})
                   end;
              fill_result_list(operator-43,0)
            end
       else {special function}
       if (operator > 127) and (operator <= 141)
```

```
      then fill_result_list(operator-57,0)
      else if (operator > 141) and (operator <= 151)
      then fill_result_list(operator-40,0)
      else stop(22)
    end
  end {production_of_object_program};

  function  thenelse: boolean;                                      {ZN}
  begin if (store[tlsc-1] mod 255 = 83{then})
        or (store[tlsc-1] mod 255 = 84{else})
    then begin tlsc:= tlsc - 2;
           fill_future_list(flib+store[tlsc],rlsc);
           unload_t_list_element(eflag);
           thenelse:= true
         end
    else thenelse:= false
  end {thenelse};

  procedure empty_t_list_through_thenelse;                          {FR}
  begin oflag:= 1;
    repeat production_of_object_program(1)
    until not thenelse
  end {empty_t_list_through_thenelse};

  function do_in_t_list: boolean;                                   {ER}
  begin if store[tlsc-1] mod 255 = 86
   then begin tlsc:= tlsc - 5;
          nlsc:= store[tlsc+2]; bn:= bn - 1;
          fill_future_list(flib+store[tlsc+1],rlsc+1);
          fill_result_list(1,88604672{2T 0X0 A}+store[tlsc]);
          do_in_t_list:= true
        end
   else do_in_t_list:= false
  end {do_in_t_list};

  procedure look_for_name;                                          {HZ}
  label 1,2;
  var i,w: integer;
  begin i:= nlib + nlsc;
  1: w:= store[i-2];
    if w = inw
    then if w mod 8 = 0
         then {at most 4 letters/digits} goto 2
         else {more than 4 letters/digits}
              if store[i-3] = fnw then goto 2;
    if w mod 8 = 0 then i:= i - 2 else i:= i - 3;
    if i > nlib then goto 1;
    stop(7);
  2: nid:= i - nlib - 1; id:= store[i-1];
    pflag:= id div d18 mod 2;
    jflag:= id div d17 mod 2;
    fflag:= id div d16 mod 2
  end {look_for_name};

  procedure look_for_constant;                                      {FW}
  var i: integer;
  begin if klib + klsc + dflag >= nlib
    then begin {move name list}
           if nlib + nlsc + 16 >= plib then stop(5);
           for i:= nlsc - 1 downto 0 do
             store[nlib+i+16]:= store[nlib+i];
           nlib:= nlib + 16
         end;
    if dflag = 0
    then begin {search integer constant}
           store[klib+klsc]:= inw;
           i:= 0;
           while store[klib+i] <> inw do i:= i + 1
         end
    else begin {search floating constant}
           store[klib+klsc]:= fnw; store[klib+klsc+1]:= inw;
           i:= 0;
           while (store[klib+i] <> fnw)
             or (store[klib+i+1] <> inw) do i:= i + 1
         end;
    if i = klsc
    then {first occurrence} klsc:= klsc + dflag + 1;
    id:= 3 * d24 + i;
    if dflag = 0 then id:= id + d19;
    jflag:= 0; pflag:= 0; fflag:= 0
  end {look_for_constant};

begin {body of main scan}                                           {EL}
  1: read_until_next_delimiter;
  2: if nflag <> 0
     then if kflag = 0
          then look_for_name
          else look_for_constant
     else begin jflag:= 0; pflag:= 0; fflag:= 0 end;
  3: if dl <= 65 then goto 64; {+,-}                                {EH}
```

```
    if dl <= 68 then goto 66; {*,/,_:}
    if dl <= 69 then goto 69; {|^}
    if dl <= 75 then goto 70; {<,_<,=,_>,>,|=}
    if dl <= 80 then goto 76; {~,^,‘,=>,_=}
    case dl of
     81: goto  81; {goto}                                          {KR}
     82: goto  82; {if}                                            {EY}
     83: goto  83; {then}                                          {EN}
     84: goto  84; {else}                                          {FZ}
     85: goto  85; {for}                                           {FE}
     86: goto  86; {do}                                            {FL}
     87: goto  87; {,}                                             {EK}
     90: goto  90; {:}                                             {FN}
     91: goto  91; {;}                                             {FS}
     92: goto  92; {:=}                                            {EZ}
     94: goto  94; {step}                                          {FH}
     95: goto  95; {until}                                         {FK}
     96: goto  96; {while}                                         {FF}
     98: goto  98; {(}                                             {EW}
     99: goto  99; {)}                                             {EU}
    100: goto 100; {[}                                             {EE}
    101: goto 101; {]}                                             {EF}
    102: goto 102; {|<}                                            {KS}
    104: goto 104; {begin}                                         {LZ}
    105: goto 105; {end}                                           {FS}
    106: goto 106; {own}                                           {KH}
    107: goto 107; {Boolean}                                       {KZ}
    108: goto 108; {integer}                                       {KZ}
    109: goto 109; {real}                                          {KE}
    110: goto 110; {array}                                         {KF}
    111: goto 111; {switch}                                        {HE}
    112: goto 112; {procedure}                                     {HY}
    end {case};

64: {+,-}                                                          {ES}
    if oflag = 0
    then begin production_of_object_program(9);
           fill_t_list_with_delimiter
         end
    else if dl = 65{-}
         then begin oh:= 10; dl:= 132{NEG};
                fill_t_list_with_delimiter
              end;
    goto 1;
```

```
66: {*,/,_:}                                                    {ET}
    production_of_object_program(10);
    fill_t_list_with_delimiter;
    goto 1;

69: {|ˆ}                                                        {KT}
    production_of_object_program(11);
    fill_t_list_with_delimiter;
    goto 1;

70: {<,_<,=,_>,>,|=}                                            {KK}
    oflag:= 1;
    production_of_object_program(8);
    fill_t_list_with_delimiter;
    goto 1;

76: {˜,ˆ,‘,=>,_=}                                               {KL}
    if dl = 76{˜}
    then begin oh:= 83-dl; goto 8202 end;
    production_of_object_program(83-dl);
    fill_t_list_with_delimiter;
    goto 1;

81: {goto}                                                      {KR}
    reservation_of_arrays; goto 1;

82:   {if}                                                      {EY}
      if eflag = 0 then reservation_of_arrays;
      fill_t_list(eflag); eflag:= 1;
8201: oh:= 0;
8202: fill_t_list_with_delimiter;
      oflag:= 1; goto 1;

83:   {then}                                                    {EN}
      repeat production_of_object_program(1) until not thenelse;
      tlsc:= tlsc - 1; eflag:= store[tlsc-1];
      fill_result_list(30{CAC},0);
      fill_result_list(2,88178688+flsc) {N 2T 'flsc'};
8301: fill_t_list(flsc); flsc:= flsc + 1;
      goto 8201;

84:   {else}                                                    {FZ}
      production_of_object_program(1);
      if store[tlsc-1] mod d8 = 84{else}
      then if thenelse then goto 84;
```

```
8401: if do_in_t_list then goto 8401;
      if store[tlsc-1] = 161 {block-begin marker}
      then begin tlsc:= tlsc - 3;
             nlsc:= store[tlsc+1];
             fill_future_list(flib+store[tlsc],rlsc+1);
             fill_result_list(12{RET},0);
             bn:= bn - 1; goto 8401
           end;
      fill_result_list(2,88080384+flsc) {2T 'flsc'};
      if thenelse {finds 'then'!}
      then tlsc:= tlsc + 1 {keep eflag in t_list};
      goto 8301;

85:   {for}                                                      {FE}
      reservation_of_arrays;
      fill_result_list(2,88080384+flsc) {2T 'flsc'};
      fora:= flsc; flsc:= flsc + 1;
      fill_t_list(rlsc);
      vflag:= 1; bn:= bn + 1;
8501: oh:= 0; fill_t_list_with_delimiter;
      goto 1;

86:   {do}                                                       {FL}
      empty_t_list_through_thenelse;
      goto 8701; {execute part of DDEL ,}
8601: {returned from DDEL ,}
     vflag:= 0; tlsc:= tlsc - 1;
     fill_result_list(2,20971520+flsc) {2S 'flsc'};
     fill_t_list(flsc); flsc:= flsc + 1;
     fill_result_list(27{FOR8},0);
     fill_future_list(flib+fora,rlsc);
     fill_result_list(19{FOR0},0);
     fill_result_list(1,88604672{2T 0X0 A}+store[tlsc-2]);
     fill_future_list(flib+forc,rlsc);
     eflag:= 0; intro_new_block1;
     goto 8501;

87:  {,}                                                         {EK}
     oflag:= 1;
     if iflag = 1
     then begin {subscript separator:}
            repeat production_of_object_program(1)
            until not thenelse;
            goto 1
          end;
      if vflag = 0 then goto 8702;
      {for-list separator:}
      repeat production_of_object_program(1)
      until not thenelse;
8701: if store[tlsc-1] mod d8 = 85{for}
      then fill_result_list(21{for2},0)
      else begin tlsc:= tlsc - 1;
             if store[tlsc] mod d8 = 96{while}
             then fill_result_list(23{for4},0)
             else fill_result_list(26{for7},0)
           end;
      if dl = 86{do} then goto 8601;
      goto 1;
8702: if mflag = 0 then goto 8705;
      {actual parameter separator:}
      if store[tlsc-1] mod d8 = 87{,}
      then if aflag = 0
           then if (store[tlsc-2] = rlsc)
                   and (fflag = 0) and (jflag = 0) and (nflag = 1)
                then begin if nid > nlscop
                       then begin if (pflag = 1) and (fflag = 0)
                              then {non-formal procedure:}
                                   test_first_occurrence;
                              {PORD construction:}
                              if (id div d15) mod 2 = 0
                              then begin {static addressing}
                                     pstb:= ((id div d24) mod d2) * d24
                                            + id mod d15;
                                     if (id div d24) mod d2 = 2
                                     then pstb:= pstb + d17
                                   end
                              else begin{dynamic addressing}
                                     pstb:= d16 + (id mod d5) * d22
                                            + (id div d5) mod d10;
                                     if (id div d16) mod 2 = 1
                                     then begin store[tlsc-2]:= pstb + d17;
                                            goto 8704
                                          end
                                   end;
                              if (id div d18) mod 2 = 1
                              then store[tlsc-2]:= pstb + d20
                              else if (id div d19) mod 2 = 1
                              then store[tlsc-2]:= pstb + d19
                              else store[tlsc-2]:= pstb;
                              goto 8704
```

```
                            end
                       else begin fill_result_list(98{TFP},0);
                              goto 8703
                            end
                     end
                else goto 8703
           else begin {completion of implicit subroutine:}
                 store[tlsc-2]:= store[tlsc-2] + d19 + d20 + d24;
                 fill_result_list(13{EIS},0); goto 8704
                end;
8703: {completion of implicit subroutine:}
      repeat production_of_object_program(1)
      until not (thenelse or do_in_t_list);
      store[tlsc-2]:= store[tlsc-2] + d20 + d24;
      fill_result_list(13{EIS},0);
8704: if dl = 87{,} then goto 9804 {prepare next parameter};
      {production of PORDs:}
      psta:= 0; unload_t_list_element(pstb);
      while pstb mod d8 = 87{,} do
      begin psta:= psta + 1; unload_t_list_element(pstb);
        if pstb div d16 mod 2 = 0
        then fill_result_list(pstb div d24, pstb mod d24)
        else fill_result_list(0,pstb);
        unload_t_list_element(pstb)
      end;
      tlsc:= tlsc - 1;
      fill_future_list(flib+store[tlsc],rlsc);
      fill_result_list(0,4718592+psta) {2A 'psta' A};
      bn:= bn - 1;
      unload_t_list_element(fflag); unload_t_list_element(eflag);
      production_transmark;
      aflag:= 0;
      unload_t_list_element(mflag); unload_t_list_element(vflag);
      unload_t_list_element(iflag); goto 1;
8705: empty_t_list_through_thenelse;
      if sflag = 0 then {array declaration} goto 1;
      {switch declaration:}
      oh:= 0; dl:= 160;
      fill_t_list(rlsc); fill_t_list_with_delimiter; goto 1;

 90: {:}                                                            {FN}
     if jflag = 0
     then begin {array declaration}
            ic:= ic + 1;
            empty_t_list_through_thenelse
```

```
         end
    else begin {label declaration}
           reservation_of_arrays;
           label_declaration
         end;
    goto 1;

91: goto 105{end};

92: {:=}                                                           {EZ}
    reservation_of_arrays;
    dl:= 128{ST}; oflag:= 1;
    if vflag = 0
    then begin if sflag = 0
           then begin {assignment statement}
                  if eflag = 0
                  then eflag:= 1
                  else dl:= 129{STA};
                  oh:= 2;
                  if pflag = 0
                  then begin {assignment to variable}
                         if nflag <> 0
                         then {assignment to scalar} generate_address;
                       end
                  else begin {assignment to function identifier}
                         dl:= dl + 2{STP or STAP};
                         fill_t_list((id div d19) mod d5{bn from id})
                       end;
                  fill_t_list_with_delimiter
                end
           else begin {switch declaration}
                  fill_result_list(2,88080384+flsc) {2T 'flsc'};
                  fill_t_list(flsc); flsc:= flsc + 1;
                  fill_t_list(nid);
                  oh:= 0; fill_t_list_with_delimiter;
                  dl:= 160;
                  fill_t_list(rlsc); fill_t_list_with_delimiter
                end
         end
    else begin {for statement}
           eflag:= 1;
           if nflag <> 0 then {simple variable} generate_address;
           fill_result_list(20{FOR1},0);
           forc:= flsc;
           fill_result_list(2,88080384+flsc) {2T 'flsc'};
           flsc:= flsc + 1;
           fill_future_list(flib+fora,rlsc);
           fill_result_list(0,4718592{2A 0 A});
           fora:= flsc;
           fill_result_list(2,71303168+flsc) {2B 'flsc};
           flsc:= flsc + 1;
           fill_result_list(9{ETMP},0)
         end;
      goto 1;

94: {step}                                                         {FH}
    empty_t_list_through_thenelse;
    fill_result_list(24{FOR5},0);
    goto 1;

95: {until}                                                        {FK}
    empty_t_list_through_thenelse;
    fill_result_list(25{FOR6},0);
    goto 8501;

96: {while}                                                        {FF}
    empty_t_list_through_thenelse;
    fill_result_list(22{FOR3},0);
    goto 8501;

98:   {(}                                                          {EW}
      oflag:= 1;
      if pflag = 1 then goto 9803;
9801: {parenthesis in expression:}
      fill_t_list(mflag);
      mflag:= 0;
9802: oh:= 0; fill_t_list_with_delimiter;
      goto 1;
9803: {begin of parameter list:}
      procedure_statement;
      fill_result_list(2,88080384+flsc) {2T 'flsc'};
      fill_t_list(iflag); fill_t_list(vflag);
      fill_t_list(mflag); fill_t_list(eflag);
      fill_t_list(fflag); fill_t_list(flsc);
      iflag:= 0; vflag:= 0; mflag:= 1; eflag:= 1;
      flsc:= flsc + 1; oh:= 0; bn:= bn + 1;
      fill_t_list_with_delimiter;
      dl:= 87{,};
9804: {prepare parsing of actual parameter:}
      fill_t_list(rlsc);
```

```
       aflag:= 0; goto 9802;

 99: {()}                                                       {EU}
     if mflag = 1 then goto 8702;
     repeat production_of_object_program(1)
     until not thenelse;
     tlsc:= tlsc - 1; unload_t_list_element(mflag);
     goto 1;

100: {[}                                                        {EE}
     if eflag = 0 then reservation_of_arrays;
     oflag:= 1; oh:= 0;
     fill_t_list(eflag); fill_t_list(iflag);
     fill_t_list(mflag); fill_t_list(fflag);
     fill_t_list(jflag); fill_t_list(nid);
     eflag:= 1; iflag:= 1; mflag:= 0;
     fill_t_list_with_delimiter;
     if jflag = 0 then generate_address {of storage function};
     goto 1;

101: {]}                                                        {EF}
     repeat production_of_object_program(1)
     until not thenelse;
     tlsc:= tlsc - 1;
     if iflag = 0
     then begin {array declaration:}
            fill_result_list(0,21495808+aic{2S 'aic' A});
            fill_result_list(90{RSF}+ibd,0) {RSF or ISF};
            arrb:= d15 + d25 + d26;
            if ibd = 1 then arrb:= arrb + d19;
            arra:= nlib + nlsc;
            repeat store[arra-1]:= arrb + pnlv;
              if store[arra-2] mod d3 = 0
              then arra:= arra - 2 else arra:= arra - 3;
              pnlv:= pnlv + (ic + 3) * d5; aic:= aic - 1
            until aic = 0;
            read_until_next_delimiter;
            if dl <> 91 then goto 1103;
            eflag:= 0; goto 1
          end;
     unload_t_list_element(nid); unload_t_list_element(jflag);
     unload_t_list_element(fflag); unload_t_list_element(mflag);
     unload_t_list_element(iflag); unload_t_list_element(eflag);
     if jflag = 0
     then begin {subscripted variable:}
```

```
            aflag:= 1; fill_result_list(56{IND},0);
            goto 1
          end;
     {switch designator:}
     nflag:= 1; fill_result_list(29{SSI},0);
     read_next_symbol;
     id:= store[nlib+nid];
     pflag:= 0; goto 3;

102: {|<}                                                         {KS}
     qc:= 1; qb:= 0; qa:= 1;
     repeat read_next_symbol;
       if dl = 102{|<} then qc:= qc + 1;
       if dl = 103{|>} then qc:= qc - 1;
       if qc > 0
       then begin qb:= qb + dl * qa; qa:= qa * d8;
              if qa = d24
              then begin fill_result_list(0,qb); qb:= 0; qa:= 1 end
            end
     until qc = 0;
     fill_result_list(0,qb+255{end marker}*qa);
     oflag:= 0; goto 1;

104: {begin}                                                      {LZ}
     if store[tlsc-1] <> 161 {block-begin marker}
     then reservation_of_arrays;
     goto 8501;

105: {end}                                                        {FS}
     reservation_of_arrays;
     repeat empty_t_list_through_thenelse
     until not do_in_t_list;
     if sflag = 0
     then begin if store[tlsc-1] = 161 {blok-begin marker}
            then begin tlsc:= tlsc - 3;
                   nlsc:= store[tlsc+1];
                   fill_future_list(flib+store[tlsc],rlsc+1);
                   fill_result_list(12{RET},0);
                   bn:= bn - 1;
                   goto 105
                 end
          end
     else begin {end of switch declaration}
            sflag:= 0;
            repeat tlsc:= tlsc - 2;
```

```
              fill_result_list(1,88604672+store[tlsc])
               {2T 'stacked RLSC' A}
            until store[tlsc-1] <> 160{switch comma};
            tlsc:= tlsc - 1; unload_t_list_element(nid);
            label_declaration;
            fill_result_list(0,85983232+48) {1T 16X1};
            tlsc:= tlsc - 1;
            fill_future_list(flib+store[tlsc],rlsc)
          end;
     eflag:= 0;
     if dl <> 105{end} then goto 1;
     tlsc:= tlsc - 1;
     if tlsc = tlib + 1 then goto 1052;
     repeat read_next_symbol
     until (dl = 91{;}) or (dl = 84{else}) or (dl = 105{end});
     jflag:= 0; pflag:= 0; fflag:= 0; nflag:= 0;
     goto 2;

106: {own}                                                          {KH}
     new_block_by_declaration;
     read_next_symbol;
     if dl = 109{real} then ibd:= 0 else ibd:= 1;
     read_until_next_delimiter;
     if nflag = 0 then goto 1102;
     goto 1082;

107: {Boolean}                                                      {KZ}
     goto 108{integer};

108:  {integer}                                                     {KZ}
      ibd:= 1;
      new_block_by_declaration;
      read_until_next_delimiter;
1081: if nflag = 0
      then begin if dl = 110{array} then goto 1101;
             goto 112{procedure}
           end;
      {scalar:}
      if bn <> 0 then goto 1083;
1082: {static addressing}
      id:= gvc;
      if ibd = 1
      then begin id:= id + d19; gvc:= gvc + 1 end
      else gvc:= gvc + 2;
      fill_name_list;
      if dl = 87{,}
      then begin read_until_next_delimiter;
             goto 1082
           end;
      goto 1;
1083: {dynamic addressing}
      id:= pnlv + d15;
      if ibd = 1
      then begin id:= id + d19;
             pnlv:= pnlv + 32; lvc:= lvc + 1
           end
      else begin pnlv:= pnlv + 2 * 32; lvc:= lvc + 2 end;
      fill_name_list;
      if dl = 87{,}
      then begin read_until_next_delimiter;
             goto 1083
           end;
      read_until_next_delimiter;
      if (dl <= 106{own}) or (dl > 109{real})
      then begin reservation_of_local_variables;
             goto 2
           end;
      if dl = 109{real} then ibd:= 0 else ibd:= 1;
      read_until_next_delimiter;
      if nflag = 1 then goto 1083 {more scalars};
      reservation_of_local_variables;
      if dl = 110{array} then goto 1101;
      goto 3;

109: {real}                                                         {KE}
     ibd:= 0;
     new_block_by_declaration;
     read_until_next_delimiter;
     if nflag = 1 then goto 1081;
     goto 2;

110:  {array}                                                       {KF}
      ibd:= 0;
      new_block_by_declaration;
1101: if bn <> 0 then goto 1103;
1102: {static bounds, constants only:}
      id:= 3 * d24;
      if ibd <> 0 then id:= id + d19;
      repeat arra:= nlsc; arrb:= tlsc;
        repeat {read identifier list:}
```

```
  read_until_next_delimiter; fill_name_list
until dl = 100{[};
arrc:= 0;
fill_t_list(2-ibd); {delta[0]}
repeat {read bound-pair list:}
  {lower bound:}
  read_until_next_delimiter;
  if dl <> 90 {:}
  then if dl = 64{+}
       then begin read_until_next_delimiter;
              arrd:= inw
            end
       else begin read_until_next_delimiter;
              arrd:= - inw
            end
  else arrd:= inw;
  arrc:= arrc - (arrd * store[tlsc-1]) mod d26;
  {upper bound:}
  read_until_next_delimiter;
  if nflag = 0
  then if dl = 65{-}
       then begin read_until_next_delimiter;
              arrd:= - inw - arrd
            end
       else begin read_until_next_delimiter;
              arrd:= inw - arrd
            end
  else arrd:= inw - arrd;
  if dl = 101{[}
  then fill_t_list(- ((arrd + 1) * store[tlsc-1]) mod d26)
  else fill_t_list(((arrd + 1) * store[tlsc-1]) mod d26)
until dl = 101{]};
arrd:= nlsc;
repeat {construction of storage function in constant list:}
  store[nlib+arrd-1]:= store[nlib+arrd-1] + klsc;
  fill_constant_list(gvc); fill_constant_list(gvc+arrc);
  tlsc:= arrb;
  repeat fill_constant_list(store[tlsc]);
    tlsc:= tlsc + 1
  until store[tlsc-1] <= 0;
  gvc:= gvc - store[tlsc-1]; tlsc:= arrb;
  if store[nlib+arrd-2] mod d3 = 0
  then arrd:= arrd - 2 else arrd:= arrd - 3
until arrd = arra;
read_until_next_delimiter
```

```
      until dl <> 87{,};
      goto 91{;};
1103: {dynamic bounds,arithmetic expressions:}
      ic:= 0; aic:= 0; id:= 0;
      repeat aic:= aic + 1;
        read_until_next_delimiter;
        fill_name_list
      until dl <> 87{,};
      eflag:= 1; oflag:= 1;
      goto 8501;

111: {switch}                                                        {HE}
     reservation_of_arrays;
     sflag:= 1;
     new_block_by_declaration;
     goto 1;

112:  {procedure}                                                    {HY}
      reservation_of_arrays;
      new_block_by_declaration;
      fill_result_list(2,88080384+flsc) {2T 'flsc'};
      fill_t_list(flsc); flsc:= flsc + 1;
      read_until_next_delimiter; look_for_name;
      label_declaration; intro_new_block;
      new_block_by_declaration1;
      if dl = 91{;} then goto 1;
      {formal parameter list:}
      repeat read_until_next_delimiter; id:= pnlv + d15 + d16;
        fill_name_list; pnlv:= pnlv + 2 * d5 {reservation PARD}
      until dl <> 87;
      read_until_next_delimiter; {for ; after )}
1121: read_until_next_delimiter;
      if nflag = 1 then goto 2;
      if dl = 104{begin} then goto 3;
      if dl <> 115{value} then goto 1123 {specification part};
      {value part:}
      spe:= d26; {value flag}
1122: repeat read_until_next_delimiter; look_for_name;
        store[nlib+nid]:= store[nlib+nid] + spe
      until dl <> 87;
      goto 1121;
1123: {specification part:}
      if (dl = 113{string}) or (dl = 110{array})
      then begin spe:= 0; goto 1122 end;
      if (dl = 114{label}) or (dl = 111{switch})
      then begin spe:= d17; goto 1122 end;
      if dl = 112{procedure}
      then begin spe:= d18; goto 1122 end;
      if dl = 109{real}
      then spe:= 0 else spe:= d19;
      if (dl <= 106) or (dl > 109) then goto 3; {if,for,goto}
      read_until_next_delimiter; {for delimiter following real/integer/boolean}
      if dl = 112{procedure}
      then begin spe:= d18; goto 1122 end;
      if dl = 110{array} then goto 1122;
1124: look_for_name; store[nlib+nid]:= store[nlib+nid] + spe;
      if store[nlib+nid] >= d26
      then begin id:= store[nlib+nid] - d26;
             id:= (id div d17) * d17 + id mod d16;
             store[nlib+nid]:= id;
             address_to_register; {generates 2S 'PARD position' A}
             if spe = 0
             then fill_result_list(14{TRAD},0)
             else fill_result_list(16{TIAD},0);
             address_to_register; {generates 2S 'PARD position' A}
             fill_result_list(35{TFR},0);
             fill_result_list(85{ST},0)
           end;
      if dl = 87{,}
      then begin read_until_next_delimiter;
             goto 1124
           end;
      goto 1121;

1052:
end {main_scan};


procedure program_loader;                                           {RZ}
var i,j,ll,list_address,id,mcp_count,crfa: integer;
    heptade_count,parity_word,read_location,stock: integer;
    from_store: 0..1;
    use: boolean;

  function logical_sum(n,m: integer): integer;
  {emulation of a machine instruction}
  var i,w: integer;
  begin w:= 0;
    for i:= 0 to 26 do
    begin w:= w div 2;
```

```
      if n mod 2 = m mod 2 then w:= w + d26;
      n:= n div 2; m := m div 2
    end;
    logical_sum:= w
  end {logical_sum};

  procedure complete_bitstock;                                    {RW}
  var i,w: integer;
  begin while bitcount > 0 {i.e., at most 20 bits in stock} do
    begin heptade_count:= heptade_count + 1;
      case from_store of
      0: {bit string read from store:}
         begin if heptade_count > 0
           then begin bitcount:= bitcount + 1;
                  heptade_count:= - 3;
                  read_location:= read_location - 1;
                  stock:= store[read_location];
                  w:= stock div d21;
                  stock:= (stock mod d21) * 64
                end
           else begin w:= stock div d20;
                  stock:= (stock mod d20) * 128
                end
         end;
      1: {bit string read from tape:}
         begin read(lib_tape,w);
           if heptade_count > 0
           then begin {test parity of the previous 4 heptades}
                  bitcount:= bitcount + 1;
                  parity_word:=
                    logical_sum(parity_word,parity_word div d4)
                    mod d4;
                  if parity_word in [0,3,5,6,9,10,12,15]
                  then stop(105);
                  heptade_count:= -3; parity_word:= w;
                  w:= w div 2
                end
           else parity_word:= logical_sum(parity_word,w)
         end
      end {case};
      for i:= 1 to bitcount - 1 do w:= 2 * w;
      bitstock:= bitstock + w; bitcount:= bitcount - 7
    end {while}
  end {complete_bitstock};
```

```
function read_bit_string(n: integer): integer;                      {RW}
var i,w: integer;
begin w:= 0;
  for i:= 1 to n do
  begin w:= 2 * w + bitstock div d26;
    bitstock:= (bitstock mod d26) * 2
  end;
  read_bit_string:= w; bitcount:= bitcount + n;
  complete_bitstock
end {read_bit_string};

procedure prepare_read_bit_string1;
var i: integer;
begin for i:= 1 to 27 - bitcount do bitstock:= 2 * bitstock;
  bitcount:= 21 - bitcount; heptade_count:= 0;
  from_store:= 0; complete_bitstock
end {prepare_read_bit_string1};

procedure prepare_read_bit_string2;
begin bitstock:= 0; bitcount:= 21; heptade_count:= 0;
  from_store:= 0; complete_bitstock;
  repeat until read_bit_string(1) = 1
end {prepare_read_bit_string2};

procedure prepare_read_bit_string3;
var w: integer;
begin from_store:= 1; bitstock:= 0; bitcount:= 21;
  repeat read(lib_tape,w) until w <> 0;
  if w <> 30 {D} then stop(106);
  heptade_count:= 0; parity_word:= 1;
  complete_bitstock;
  repeat until read_bit_string(1) = 1
end {prepare_read_bit_string3};

function address_decoding: integer;                                 {RY}
var w,a,n: integer;
begin w:= bitstock;
  if w < d26 {code starts with 0}
  then begin {0}      n:= 1; a:= 0; w:= 2 * w end
  else begin {1xxxxx} n:= 6; a:= (w div d21) mod d5;
         w:= (w mod d21) * d6
       end;
  if w < d25 {00}
  then begin {00} n:= n + 2; a:= 32 * a + 0; w:= w * 4 end else
  if w < d26 {01}
  then begin {01xx} n:= n + 4; a:= 32 * a + w div d23;
         if a mod d5 < 6
         then {010x} a:= a - 3 else {011x} a:= a - 2;
         w:= (w mod d23) * d4
       end
  else begin {1xxxxx} n:= n + 6;
         a:= a * 32 + (w div d21) mod d5;
         w:= (w mod d21) * d6
       end;
  if w < d25 {00}
  then begin {00} n:= n + 2; a:= 32 * a + 1 end else
  if w < d26 {01}
  then begin {01x} n:= n + 3; a := 32 * a + w div d24 end
  else begin {1xxxxx} n:= n + 6;
         a:= 32 * a + (w div d21) mod d5
       end;
  w:= read_bit_string(n); address_decoding:= a
end {address_decoding};

function read_mask: integer;                                      {RN}
var c: 0 .. 19;
begin
  if bitstock < d26 {code starts with 0}
  then {0x} c:= read_bit_string(2) else
  if bitstock < d26 + d25 {01}
  then {10x} c:= read_bit_string(3) - 2
  else {11xxxx} c:= read_bit_string(6) - 44;
  case c of
     0: read_mask:=   656; {0,   2S 0    A  }
     1: read_mask:= 14480; {3,   2B 0    A  }
     2: read_mask:= 10880; {2,   2T 0 X0    }
     3: read_mask:=  2192; {0,   2B 0    A  }
     4: read_mask:=   144; {0,   2A 0    A  }
     5: read_mask:= 10368; {2,   2B 0 X0    }
     6: read_mask:=  6800; {1,   2T 0    A  }
     7: read_mask:=     0; {0,   0A 0 X0    }
     8: read_mask:= 12304; {3,   0A 0    A  }
     9: read_mask:= 10883; {2, N 2T 0 X0    }
    10: read_mask:=  6288; {1,   2B 0    A  }
    11: read_mask:=  4128; {1,   0A 0 X0 B  }
    12: read_mask:=  8832; {2,   2S 0 X0    }
    13: read_mask:=   146; {0, Y 2A 0    A  }
    14: read_mask:=   256; {0,   4A 0 X0    }
    15: read_mask:=   134; {0, Y 2A 0 X0   P}
    16: read_mask:=   402; {0, Y 6A 0    A  }
```

```
      17: read_mask:=  4144; {1,   0A 0 X0 C  }
      18: read_mask:=    16; {0,   0A 0    A  }
      19: read_mask:= address_decoding
    end {case}
  end {read_mask};

  function read_binary_word: integer;                               {RF}
  var w: integer; opc: 0 .. 3;
  begin if bitstock < d26 {code starts with 0}
    then begin {OPC >= 8}
           if bitstock < d25 {00}
           then if bitstock < d24 {000}
                then w:= 4 {code is 000x}
                else w:= 5 {code is 001xx}
           else if bitstock < d25 + d24 {010}
                then if bitstock < d25 + d23 {0100}
                     then w:= 6 {0100xx}
                     else w:= 7 {0101xxx}
                else w:= 10 {011xxxxxxx};
           w:= read_bit_string(w);
           if w <  2 {000x}    then {no change} else
           if w <  8 {001xx}   then w:= w -   2 else
           if w < 24 {010xx}   then w:= w -  10 else
           if w < 48 {0101xxx} then w:= w -  30
                else {011xxxxxxx}   w:= w - 366;
           read_binary_word:= opc_table[w]
         end {0}
    else begin w:= read_bit_string(1);
           w:= read_mask; opc:= w div d12;
           w:= (w mod d12) * d15 + address_decoding;
           case opc of
             0: ;
             1: w:= w + list_address;
             2: begin if w div d17 mod 2 = 1 {d17 = 1}
                  then w:= w - d17
                  else w:= w + d19;
                  w:= w  - w mod d15 + store[flib + w mod d15]
                end;
             3: if klib = crfb
                then w:= w - w mod d15 + store[mlib+w mod d15]
                else w:= w + klib
           end {case};
           read_binary_word:= w
         end {1}
  end {read_binary_word};
```

```
    procedure test_bit_stock;                                           {RH}
    begin if bitstock <> 63 * d21 then stop(107)
    end {test_bit_stock};

    procedure typ_address(a: integer);                                  {RT}
    begin writeln(output);
      write(output,a div 1024:2,' ',(a mod 1024) div 32:2,' ',a mod 32:2)
    end {typ_address};

    procedure read_list;                                                {RL}
    var i,j,w: integer;
    begin for i:= ll - 1 downto 0 do
      begin w:= read_binary_word;
        if list_address + i <= flib + flsc
        then begin {shift FLI downwards}
               if flib <= read_location
               then stop(98);
               for j:= 0 to flsc - 1 do
               store[read_location+j]:= store[flib+j];
               flib:= read_location
             end;
        store[list_address+i]:= w
      end {for i};
      test_bit_stock;
    end {read_list};

    function read_crf_item: integer;                                    {RS}
    begin if crfa mod 2 = 0
      then read_crf_item:= store[crfa div 2] div d13
      else read_crf_item:= store[crfa div 2] mod d13;
      crfa:= crfa + 1
    end {read_crf_item};

  begin {of program loader}
    rlib:= (klie - rlsc - klsc) div 32 * 32;
  {increment entries in future list:}
    for i:= 0 to flsc - 1 do store[flib+i]:= store[flib+i] + rlib;
  {move KLI to final position:}
    for i:= klsc - 1 downto 0 do store[rlib+rlsc+i]:= store[klib+i];
    klib:= rlib + rlsc;
  {prepare mcp-need analysis:}
    mcpe:= rlib; mcp_count:= 0;
    for i:= 0 to 127 do store[mlib+i]:= 0;
  {determine primary need of MCP's from name list:}
  i:= nlsc0;
  while i > nlscop do
  begin id:= store[nlib+i-1];
    if store[nlib+i-2] mod d3 = 0
    then {at most 4 letter/digit identifier} i:= i - 2
    else {at least 5 letters or digits} i:= i - 3;
    if (id div d15) mod 2 = 0
    then begin {MCP is used} mcp_count:= mcp_count + 1;
           store[mlib+(store[flib+id mod d15]-rlib) mod d15]:=
             - (flib + id mod d15)
         end
  end;
{determine secondary need using the cross-reference list:}
  crfa:= 2 * crfb;
  ll:= read_crf_item {for MCP length};
  while ll <> 7680 {end marker} do
  begin i:= read_crf_item {for MCP number};
    use:= (store[mlib+i] <> 0);
    j:= read_crf_item {for number of MCP needing the current one};
    while j <> 7680 {end marker} do
    begin use:= use or (store[mlib+j] <> 0); j:= read_crf_item end;
    if use
    then begin mcpe:= mcpe - ll;
           if mcpe <= mcpb then stop(25);
           if store[mlib+i] < 0
           then {primary need} store[-store[mlib+i]]:= mcpe
           else {only secondary need} mcp_count:= mcp_count + 1;
           store[mlib+i]:= mcpe
         end;
    ll:= read_crf_item
  end;
{load result list RLI:}
  ll:= rlsc; read_location:= rnsb;
  prepare_read_bit_string1;
  list_address:= rlib; read_list;
  if store[rlib] <> opc_table[89{START}] then stop(101);
  typ_address(rlib);
{copy MLI:}
  for i:= 0 to 127 do store[crfb+i]:= store[mlib+i];
  klib:= crfb; flsc:= 0;
{load MCP's from store:}
  prepare_read_bit_string2;
  ll:= read_bit_string(13) {for length or end marker};
  while ll < 7680 do
  begin i:= read_bit_string(13) {for MCP number};
```

```
    list_address:= store[crfb+i];
    if list_address <> 0
    then begin read_list; test_bit_stock;
           mcp_count:= mcp_count - 1;
           store[crfb+i]:= 0
         end
    else repeat read_location:= read_location - 1
         until store[read_location] = 63 * d21;
    prepare_read_bit_string2; ll:= read_bit_string(13)
  end;
{load MCP's from tape:}
  reset(lib_tape);
  while mcp_count <> 0 do
  begin writeln(output);
    writeln(output,'load (next) library tape into the tape reader');
    prepare_read_bit_string3;
    ll:= read_bit_string(13) {for length or end marker};
    while ll < 7680 do
    begin i:= read_bit_string(13) {for MCP number};
      list_address:= store[crfb+i];
      if list_address <> 0
      then begin read_list; test_bit_stock;
             mcp_count:= mcp_count - 1;
             store[crfb+i]:= 0
           end
      else repeat repeat read(lib_tape,ll) until ll = 0;
             read(lib_tape,ll)
           until ll = 0;
      prepare_read_bit_string3; ll:= read_bit_string(13)
    end
  end;
{program loading completed:}
  typ_address(mcpe)
end {program_loader};


{main program}

begin
{initialization of word_del_table}                                 {HT}
  word_del_table[10]:= 15086; word_del_table[11]:=    43;
  word_del_table[12]:=     1; word_del_table[13]:=    86;
  word_del_table[14]:= 13353; word_del_table[15]:= 10517;
  word_del_table[16]:=    81; word_del_table[17]:= 10624;
  word_del_table[18]:=    44; word_del_table[19]:=     0;
  word_del_table[20]:=     0; word_del_table[21]:= 10866;
  word_del_table[22]:=     0; word_del_table[23]:=     0;
  word_del_table[24]:=   106; word_del_table[25]:=   112;
  word_del_table[26]:=     0; word_del_table[27]:= 14957;
  word_del_table[28]:=     2; word_del_table[29]:=     2;
  word_del_table[30]:=    95; word_del_table[31]:=   115;
  word_del_table[32]:= 14304; word_del_table[33]:=     0;
  word_del_table[34]:=     0; word_del_table[35]:=     0;
  word_del_table[36]:=     0; word_del_table[37]:=     0;
  word_del_table[38]:=   107;

{initialization of flex_table}                                        {LK}
  flex_table[  0]:=    -2; flex_table[  1]:= 19969; flex_table[  2]:= 16898;
  flex_table[  3]:=    -0; flex_table[  4]:= 18436; flex_table[  5]:=    -0;
  flex_table[  6]:=    -0; flex_table[  7]:= 25863; flex_table[  8]:= 25096;
  flex_table[  9]:=    -0; flex_table[ 10]:=    -0; flex_table[ 11]:=    -1;
  flex_table[ 12]:=    -0; flex_table[ 13]:=    -1; flex_table[ 14]:= 41635;
  flex_table[ 15]:=    -0; flex_table[ 16]:= 31611; flex_table[ 17]:=    -0;
  flex_table[ 18]:=    -0; flex_table[ 19]:= 17155; flex_table[ 20]:=    -0;
  flex_table[ 21]:= 23301; flex_table[ 22]:= 25606; flex_table[ 23]:=    -0;
  flex_table[ 24]:=    -0; flex_table[ 25]:= 25353; flex_table[ 26]:= 30583;
  flex_table[ 27]:=    -0; flex_table[ 28]:=    -1; flex_table[ 29]:=    -0;
  flex_table[ 30]:=    -0; flex_table[ 31]:=    -1; flex_table[ 32]:= 19712;
  flex_table[ 33]:=    -0; flex_table[ 34]:=    -0; flex_table[ 35]:= 14365;
  flex_table[ 36]:=    -0; flex_table[ 37]:= 14879; flex_table[ 38]:= 15136;
  flex_table[ 39]:=    -0; flex_table[ 40]:=    -0; flex_table[ 41]:= 15907;
  flex_table[ 42]:=    -1; flex_table[ 43]:=    -0; flex_table[ 44]:=    -1;
  flex_table[ 45]:=    -0; flex_table[ 46]:=    -0; flex_table[ 47]:=    -1;
  flex_table[ 48]:=    -0; flex_table[ 49]:= 17994; flex_table[ 50]:= 14108;
  flex_table[ 51]:=    -0; flex_table[ 52]:= 14622; flex_table[ 53]:=    -0;
  flex_table[ 54]:=    -0; flex_table[ 55]:= 15393; flex_table[ 56]:= 15650;
  flex_table[ 57]:=    -0; flex_table[ 58]:=    -0; flex_table[ 59]:= 30809;
  flex_table[ 60]:=    -0; flex_table[ 61]:=    -1; flex_table[ 62]:= 30326;
  flex_table[ 63]:=    -0; flex_table[ 64]:= 19521; flex_table[ 65]:=    -0;
  flex_table[ 66]:=    -0; flex_table[ 67]:= 12309; flex_table[ 68]:=    -0;
  flex_table[ 69]:= 12823; flex_table[ 70]:= 13080; flex_table[ 71]:=    -0;
  flex_table[ 72]:=    -0; flex_table[ 73]:= 13851; flex_table[ 74]:=    -1;
  flex_table[ 75]:=    -0; flex_table[ 76]:=    -1; flex_table[ 77]:=    -0;
  flex_table[ 78]:=    -0; flex_table[ 79]:=    -1; flex_table[ 80]:=    -0;
  flex_table[ 81]:= 11795; flex_table[ 82]:= 12052; flex_table[ 83]:=    -0;
  flex_table[ 84]:= 12566; flex_table[ 85]:=    -0; flex_table[ 86]:=    -0;
  flex_table[ 87]:= 13337; flex_table[ 88]:= 13594; flex_table[ 89]:=    -0;
  flex_table[ 90]:=    -0; flex_table[ 91]:= 31319; flex_table[ 92]:=    -0;
  flex_table[ 93]:=    -1; flex_table[ 94]:=    -1; flex_table[ 95]:=    -0;
  flex_table[ 96]:=    -0; flex_table[ 97]:=  9482; flex_table[ 98]:=  9739;
  flex_table[ 99]:=    -0; flex_table[100]:= 10253; flex_table[101]:=    -0;
  flex_table[102]:=    -0; flex_table[103]:= 11024; flex_table[104]:= 11281;
  flex_table[105]:=    -0; flex_table[106]:=    -0; flex_table[107]:= 31832;
  flex_table[108]:=    -0; flex_table[109]:=    -1; flex_table[110]:=    -1;
  flex_table[111]:=    -0; flex_table[112]:= 31040; flex_table[113]:=    -0;
  flex_table[114]:=    -0; flex_table[115]:=  9996; flex_table[116]:=    -0;
  flex_table[117]:= 10510; flex_table[118]:= 10767; flex_table[119]:=    -0;
  flex_table[120]:=    -0; flex_table[121]:= 11538; flex_table[122]:=    -2;
  flex_table[123]:=    -0; flex_table[124]:=    -2; flex_table[125]:=    -0;
  flex_table[126]:=    -0; flex_table[127]:=    -2;

{preparation of prescan}                                            {LE}
  rns_state:= virginal; scan:= 1;
  read_until_next_delimiter;

  prescan;                                                          {HK}

  {writeln;
  for bn:= plib to plie do writeln(bn:5,store[bn]:10);
  writeln;}

{preparation of main scan:}                                         {HL}
  rns_state:= virginal; scan:= - 1;
  iflag:= 0; mflag:= 0; vflag:= 0; bn:= 0; aflag:= 0; sflag:= 0;
  eflag:= 0; rlsc:= 0; flsc:= 0; klsc:= 0; vlam:= 0;
  flib:= rnsb + 1; klib:= flib + 16; nlib:= klib + 16;
  if nlib + nlsc0 >= plib then stop(25);
  nlsc:= nlsc0; tlsc:= tlib; gvc:= gvc0;
  fill_t_list(161);
{prefill of name list:}
  store[nlib +  0]:= 27598040;
  store[nlib +  1]:=   265358;             {read}
  store[nlib +  2]:= 134217727 -        6;
  store[nlib +  3]:= 61580507;
  store[nlib +  4]:=   265359;             {print}
  store[nlib +  5]:= 134217727 -  53284863;
  store[nlib +  6]:=   265360;             {TAB}
  store[nlib +  7]:= 134217727 -  19668591;
  store[nlib +  8]:=   265361;             {NLCR}
  store[nlib +  9]:= 134217727 -         0;
  store[nlib + 10]:= 134217727 -  46937177;
  store[nlib + 11]:=   265363;             {SPACE}
  store[nlib + 12]:= 53230304;
  store[nlib + 13]:=   265364;             {stop}
  store[nlib + 14]:= 59085824;
```

```
store[nlib + 15]:=   265349;              {abs}
store[nlib + 16]:= 48768224;
store[nlib + 17]:=   265350;              {sign}
store[nlib + 18]:= 61715680;
store[nlib + 19]:=   265351;              {sqrt}
store[nlib + 20]:= 48838656;
store[nlib + 21]:=   265352;              {sin}
store[nlib + 22]:= 59512832;
store[nlib + 23]:=   265353;              {cos}
store[nlib + 24]:= 48922624;
store[nlib + 25]:=   265355;              {ln}
store[nlib + 26]:= 53517312;
store[nlib + 27]:=   265356;              {exp}
store[nlib + 28]:= 134217727 -       289;
store[nlib + 29]:= 29964985;
store[nlib + 30]:=   265357;              {entier}

store[nlib + 31]:= 134217727 -  29561343;
store[nlib + 32]:=   294912;              {SUM}
store[nlib + 33]:= 134217727 -  14789691;
store[nlib + 34]:= 134217727 -  15115337;
store[nlib + 35]:=   294913;              {PRINTTEXT}
store[nlib + 36]:= 134217727 -  27986615;
store[nlib + 37]:=   294914;              {EVEN}
store[nlib + 38]:= 134217727 -       325;
store[nlib + 39]:= 21928153;
store[nlib + 40]:=   294915;              {arctan}
store[nlib + 41]:= 134217727 -  15081135;
store[nlib + 42]:=   294917;              {FLOT}
store[nlib + 43]:= 134217727 -  14787759;
store[nlib + 44]:=   294918;              {FIXT}
store[nlib + 45]:= 134217727 -      3610;
store[nlib + 46]:= 134217727 -  38441163;
store[nlib + 47]:=   294936;              {ABSFIXT}

intro_new_block2;
bitcount:= 0; bitstock:= 0; rnsb:= bim;
fill_result_list(96{START},0);
pos:= 0;
main_scan;                                                  {EL}
fill_result_list(97{STOP},0);

{writeln; writeln('FLI:');
for bn:= 0 to flsc-1 do
writeln(bn:5,store[flib+bn]:10);}
```

```
  {writeln; writeln('KLI:');
  for bn:= 0 to klsc-1 do
  writeln(bn:5,store[klib+bn]:10,
         (store[klib+bn] mod 134217728) div 16777216 : 10,
                     (store[klib+bn] mod  16777216) div  2097152 : 2,
                     (store[klib+bn] mod   2097152) div   524288 : 3,
                     (store[klib+bn] mod    524288) div   131072 : 2,
                     (store[klib+bn] mod    131072) div    32768 : 2,
                     (store[klib+bn] mod     32768) div     1024 : 4,
                     (store[klib+bn] mod      1024) div       32 : 3,
                     (store[klib+bn] mod        32) div        1 : 3);}

 {preparation of program loader}
  opc_table[ 0]:=  33; opc_table[ 1]:=  34; opc_table[ 2]:=  16;
  opc_table[ 3]:=  56; opc_table[ 4]:=  58; opc_table[ 5]:=  85;
  opc_table[ 6]:=   9; opc_table[ 7]:=  14; opc_table[ 8]:=  18;
  opc_table[ 9]:=  30; opc_table[10]:=  13; opc_table[11]:=  17;
  opc_table[12]:=  19; opc_table[13]:=  20; opc_table[14]:=  31;
  opc_table[15]:=  35; opc_table[16]:=  39; opc_table[17]:=  61;
  opc_table[18]:=   8; opc_table[19]:=  10; opc_table[20]:=  11;
  opc_table[21]:=  12; opc_table[22]:=  15;
  for ii:= 23 to 31 do opc_table[ii]:= ii - 2;
  opc_table[32]:=  32; opc_table[33]:=  36; opc_table[34]:=  37;
  opc_table[35]:=  38;
  for ii:= 36 to 51 do opc_table[ii]:= ii + 4;
  opc_table[52]:=  57; opc_table[53]:=  59; opc_table[54]:=  60;
  for ii:= 55 to 102 do opc_table[ii]:= ii + 7;

  store[crfb+ 0]:=   30 * d13 +    0; store[crfb+ 1]:= 7680 * d13 +   20;
  store[crfb+ 2]:=    1 * d13 + 7680; store[crfb+ 3]:=   12 * d13 +    2;
  store[crfb+ 4]:= 7680 * d13 +   63; store[crfb+ 5]:=    3 * d13 + 7680;
  store[crfb+ 6]:=   15 * d13 +    4; store[crfb+ 7]:=    3 * d13 + 7680;
  store[crfb+ 8]:=  100 * d13 +    5; store[crfb+ 9]:= 7680 * d13 +  134;
  store[crfb+10]:=    6 * d13 +   24; store[crfb+11]:= 7680 * d13 +   21;
  store[crfb+12]:=   24 * d13 + 7680; store[crfb+13]:= 7680 * d13 + 7680;

  store[mcpb]:= 63 * d21; store[mcpb+1]:= 63 * d21;

  program_loader;

  writeln(output); writeln(output); writeln(output);
  for ii:= mcpe to rlib + rlsc + klsc - 1 do
  writeln(output,ii:5,store[ii]:9)
```

```
end.
```