```
begin
   comment GIER DEMONSTRATION PROGRAM−1D;
   boolean newnim, newmap, newlineq, newprime, newlanu;
   integer linerest, oldrand, type;
   switch TYPE := NIM, MAP, LINEQ, PRIME, LANU, FINISH;
   procedure NEWPAGE;
   begin
      for linerest := linerest − 1 while linerest ≥ −9 do skrvvr;
      linerest := 62
   end NEWPAGE;
   procedure LINE;
   begin
      linerest := linerest − 1;
      skrvvr;
      if linerest < 0 then NEWPAGE
   end LINE;
   procedure SHIFT(n);
   value n;
   integer n;
   if linerest < n then NEWPAGE;
   procedure CHECKLINE;
   if tegn= 64 ∨ tegn= 192 then
   begin
      linerest := linerest − 1;
      SHIFT(0)
   end CHECKLINE;
   integer procedure RANDOM(n);
   value n;
   integer n;
   begin
      real y, MOD;
      MOD := 32768;
      y := oldrand×6859;
      oldrand := y − MOD×entier(y/MOD);
      RANDOM := 1 + entier(n×oldrand/MOD)
   end RANDOM;
START: oldrand := 999;
   linerest := 65;
   newnim := newmap := newlineq := newprime := newlanu := true;
   skrvtegn(62);
   skrvtekst(‹<
                          GIER DEMONSTRATION PROGRAM 1D

     Programmet kan bruges paa 5 forskellige maader:

         1.   Tændstikspillet: NIM.
         2.   Trykning af tilfældige landkort.
         3.   Løsning af tilfældige lineære ligninger.
         4.   Beregning af primtal.
         5.   Beregning af store tal.
              6 giver program slut.
     Skriv Deres initialer her:›);
   linerest := linerest − 11;
   begin
      integer i, j, sum;
      sum := tasttegn+ tasttegn;
      for i := 1 step 1 until sum do j := RANDOM(1)
   end of advance of random procedure;
RESTART: LINE;  LINE;
   SHIFT(10);
   skrvtekst(‹<Vælg program type 1−5 (6 giver program stop): ›);
   type := tasttegn;
   LINE;  LINE;
   go to TYPE[type];
   go to RESTART;
NIM: SHIFT(10);
```

```
   skrvtekst(‹<Type 1.  Tændstikspillet:  NIM›);
   LINE;  LINE;
   begin comment NIM−block;
      boolean longtext, winmessage, wrongmessage, loosemessage, present;
      integer M, G, N, g, n, t, gno, remove, fact, R, boolsum, aritsum, ask,
              GMAX, nmax;
      longtext := newnim;
      if newnim then
      begin
         SHIFT(10);
         skrvtekst(‹<
                             SPILLEREGLER FOR NIM
     Spillet begynder med et tilfældigt udvalg af G bunker af tændstikker.  Hver
bunke indeholder højst M tændstikker.  M skrives som 2↑N − 1, og De skal opgive
N og G.  Vi vil saa skiftevis fjerne tændstikker fra bunkerne.  Den, som
fjerner den eller de sidste tændstikker, har vundet.  Kun een bunke maa røres
i hvert træk, og man skal fjerne mindst een tændstik fra den bunke.
›);
         linerest := linerest − 7
      end if newnim;
AGAIN: SHIFT(4);
      LINE;
      winmessage := wrongmessage := loose message :=  false;
      skrvtekst(‹<Opgiv N: ›);
      N := tast;
      if N > 9 then N := 9;
      LINE;
      CHECKLINE;
      M := 2↑N − 1;
      ask := 0;
      skrvtekst(‹<Opgiv G: ›);
      G := tast;
      if G > 15 then G := 15;
      LINE;
      CHECKLINE;
      LINE;  LINE;
      begin comment inner NIM−block;
         integer array GROUP[1:G], SUM[1:N], BITS[1:G, 1:N];
         procedure PRINTGROUPS;
         for g := 1 step 1 until G do skrv(‹−ndd›, GROUP[g]);
         procedure DISPLAY BITS(g);
         value g;
         integer g;
         begin
            R := GROUP[g];
            fact := 2;
            for n := 1 step 1 until N do
            begin
               present := R : fact×fact ǂ R;
               if present then R := R − fact : 2;
               BITS[g,n] := if present then 1 else 0;
               fact := fact×2
            end for n
         end DISPLAY BITS;
         procedure FIND SUM;
         begin
            aritsum := 0;
            for g := 1 step 1 until G do aritsum := aritsum + GROUP[g];
            for n := 1 step 1 until N do
            begin
               boolsum := 0;
               for g := 1 step 1 until G do boolsum := boolsum + BITS[g,n];
               SUM[n] := boolsum − boolsum : 2×2
            end;
            boolsum := 0;
            fact := 1;
            for n := 1 step 1 until N do
            begin
               boolsum := boolsum + SUM[n]×fact;
               fact := 2×fact
            end for n
         end FIND SUM;
         for g := 1 step 1 until G do
         begin
            GROUP[g] := RANDOM(M);
            DISPLAY BITS(g)
         end for g;
         FIND SUM;
         SHIFT(4);
         skrvtekst(≮<Her er bunkerne:≯);
         LINE;
         skrvtekst(≮<Bunke nr.:        ≯);
         for g := 1 step 1 until G do skrv(≮-ndd≯, g);
         LINE;
         skrvtekst(≮<Antal tændstikker:≯);
         PRINTGROUPS;
BB:      SHIFT(4);
         skrvtekst(≮<
     Hvis De ønsker at gøre det første træk, skriver De et 1-tal her, ellers
et 2-tal: ≯);
         linerest := linerest - 2;
         t := tasttegn;
         if t ǂ 1 ∧ t ǂ 2 then go to BB;
         if t = 2 then go to II;
GG:      SHIFT(4);
         LINE;
         skrvtekst(if longtext then
≮<Skriv her nummeret paa den bunke, fra hvilken De vil fjerne tændstikker: ≯
         else ≮<Vælg Deres bunke: ≯);
CC:      gno := tast;
         CHECKLINE;
         LINE;
         if gno < 1 ∨ gno > G then
         begin
            skrvtekst(
≮<Undskyld, men tallet er for ≯, if gno < 1 then ≮<lille.≯ else ≮<stort.≯);
DD:         LINE;
            skrvtekst(≮<  Prøv igen her: ≯);
            go to CC
         end if out of range;
         if GROUP[gno] = 0 then
         begin
            skrvtekst(≮<Undskyld, men denne bunke er tom.≯);
            go to DD
         end if empty group;
         remove := GROUP[gno];
         if remove ǂ 1 then
         begin
            SHIFT(4);
            skrvtekst(if longtext then
≮<Og antallet af tændstikker, De vil fjerne: ≯ else ≮<Og antallet: ≯);
EE:         remove := tast;
            CHECKLINE;
            LINE;
            if remove < 1 then
            begin
               skrvtekst(≮<De skal fjerne mindst een tændstik.  Prøv igen: ≯);
               go to EE
            end;
            if remove > GROUP[gno] then
            begin
               skrvtekst(
≮<Saa mange er der ikke i bunken.  De fjerner altsaa hele bunken.≯);
               LINE;
               remove := GROUP[gno]
            end if too many
         end if more than one;
         GROUP[gno] := GROUP[gno] − remove;
         if longtext then longtext := false;
         DISPLAY BITS(gno);
         FIND SUM;
II:      if boolsum ǂ 0 then
         begin
            SHIFT(4);
            if winmessage ∧ −, wrongmessage then
            begin
               skrvtegn(29);
               skrvtekst(ǂ<Det var forkert.  Nu kan De ikke vinde.ǂ);
FF:            loosemessage := wrongmessage := true;
               skrvtegn(62);
               LINE
            end of blunder;
            if loosemessage then
            begin
               ask := ask + 1;
               if ask : 3×3 = ask then
               begin
                  SHIFT(4);
                  skrvtekst(
ǂ<Hvis De giver fortabt, saa skriv et 1−tal her: ǂ);
                  t := tasttegn;
                  LINE;
                  if t = 1 then go to ASK FOR MORE
               end if third time;
               go to GIERMOVE
            end;
            skrvtegn(29);
            skrvtekst(ǂ<De kan ikke vinde dette spil.ǂ);
            go to FF
         end if boolsum ǂ 0;
         if aritsum = 0 then
         begin
            SHIFT(4);
            skrvtekst(ǂ<De har vundet.  Tillykke.ǂ);
            go to ASK FOR MORE
         end if finished;
         if −, winmessage then
         begin
            SHIFT(4);
            winmessage := true;
            skrvtegn(29);
            skrvtekst(ǂ<Hvis De spiller rigtigt, kan De vinde dette spil.ǂ);
            skrvtegn(62);
            LINE
         end if not winmessage;
         GMAX := GROUP[1];
         gno := 1;
         for g := 2 step 1 until G do
         begin
            if GROUP[g] > GMAX then
            begin
               GMAX := GROUP[g];
               gno := g
            end
         end search of largest group;
         remove := 1;
JJ:      GROUP[gno] := GROUP[gno] − remove;
         SHIFT(4);
         if GROUP[gno] > 0 then
         begin
            skrvtekst(ǂ<Jeg fjerner nuǂ);
            skrv(≮−ndd≯, remove);
            skrvtekst(≮< fra bunke nr.≯)
         end group not empty
         else
         skrvtekst(≮<Jeg fjerner nu hele bunke nr.≯);
         skrv(≮−ndd≯, gno);
         skrvtekst(≮<. Bunkerne indeholder nu:≯);
         LINE;
         LINE;
         PRINTGROUPS;
         DISPLAY BITS (gno);
         FIND SUM;
         if aritsum > 0 then go to GG;
         SHIFT(4);
         LINE;
         skrvtekst(≮<De har tabt.≯);
ASK FOR MORE:  LINE;
         newnim := false;
HH:      skrvtekst(≮<
Hvis De ønsker at prøve igen, saa skriv et 1−tal her, ellers et 2−tal: ≯);
         linerest := linerest − 1;
         t := tasttegn;
         if t ‡ 1 ∧ t ‡ 2 then go to HH;
         LINE;  LINE;
         go to if t = 1 then AGAIN else RESTART;
GIERMOVE: for g := 1 step 1 until G do
         begin
            if boolsum = GROUP[g] then
            begin
               remove := boolsum;
               gno := g;
               go to JJ
            end if remove whole group
         end for g;
         for n := N step −1 until 1 do
         begin
            if SUM[n] = 1 then
            begin
               nmax := n;
               go to KK
            end hit
         end for n;
KK:      for g := 1 step 1 until G do
         begin
            if BITS[g, nmax] = 1 then
            begin
               gno := g;
               remove := 0;
               fact := 1;
               for n := 1 step 1 until nmax do
               begin
                  if SUM[n] = 1 then
                  remove := remove + (if BITS[gno, n] = 1 then fact else − fact);
                  fact := fact×2
               end for n;
               go to JJ
            end if hit Bits
         end for g;
         go to HH
      end inner NIM−block
   end NIM;
MAP:
   begin comment MAP−block;
      boolean red, even;
      integer a, amin, amax, b, bs, fh, fhmin, fhmax, fv, fvmin, fvmax, h, hs,
              k, n, ncon, nmap, p, p2, p3, q, r, r1, r2, s, s1, s2, spr;
      real d, dmin, e, f, g, j, v;
      SHIFT(10);
      skrvtekst(‡<Type 2.  Trykning af tilfældige landkort.‡);
      LINE;  LINE;
      if newmap then
      begin
         SHIFT(18);
         skrvtekst(‡<
     Følgende parametre bruges i dette program:
                                                   Typiske værdier:
     nmap:    Antal landkort.                             1
     h:       Kortenes højde.                            60
     b:       Kortenes bredde.                           80
     hs:      Højde af delkort.                          15
     bs:      Bredde af delkort.                         20
     ncon:    Antal lande per delkort.                    1
     fh:      Vandret skalafaktor.                        1−5
              Minimum:  fhmin,  maximum:  fhmax.
     fv:      Lodret skalafaktor.                         1−5
              Minimum:  fvmin,  maximum:  fvmax.
     a:       Drejningsvinkel.                           45
              Minimum:  amin,  maximum:  amax.
     spr:     Spredningsfaktor:  Minimum  1, maximum  10  1
‡);
         linerest := linerest − 16;
         newmap := false
      end if newmap;
LL:   LINE;
      SHIFT(5);
      skrvtekst(‡<
     Opgiv parametrene:
 nmap  h   b   hs  bs  ncon  fhmin fhmax fvmin fvmax amin amax  spr
‡);
      nmap := tast;  h := tast;  b := tast;
      hs := tast;  bs := tast;  ncon := tast;
      fhmin := tast;  fhmax := tast;  fvmin := tast;
      fvmax := tast;  amin := tast;  amax := tast;
      spr := tast;
      r1 := h : hs;
      s1 := b : bs;
      p2 := r1×s1×ncon;
      linerest := linerest − 3;
      LINE;
      for n := 1 step 1 until nmap do
      begin comment inner MAP−block;
         integer array cx, cy, fh, fv[1:p2];
         array cosv, sinv[1:p2];
         p := 0;
         for r := 1 step 1 until r1 do
         for s := 1 step 1 until s1 do
         for q := 1 step 1 until ncon do
         begin
            p := p+1;
            cx[p] := (s−1)×bs + bs:2 + (RANDOM(bs) − bs:2):spr;
            cy[p] := (r−1)×hs + hs:2 + (RANDOM(hs) − hs:2):spr;
            fh[p] := fhmin + RANDOM(fhmax−fhmin) − 1;
            if fh[p] < 1 then fh[p] := 1;
            fv[p] := fvmin + RANDOM(fvmax−fvmin) − 1;
            if fv[p] < 1 then fv[p] := 1;
            v := 3.14159265/180×(amin + RANDOM(amax−amin));
            cosv[p] := cos(v);
            sinv[p] := sin(v)
         end for q, s, and r;
         SHIFT(h+3);
         LINE;
         skrvtegn(62);
         red := false;
         for r := 1 step 1 until h do
         begin
            for s := 1 step 1 until b do
            begin
               dmin := 1₁₀5;
               for p := 1 step 1 until p2 do
               begin
                  e := cx[p];
                  f := cy[p];
                  g := cosv[p];
                  j := sinv[p];
                  d := (((e-s)×g - (f-r)×j)/fv[p])↑2
                       + (((e-s)×j + (f-r)×g)/fh[p])↑2;
                  if d < dmin then
                  begin
                     dmin := d;
                     p3 := p
                  end if lower distance
               end for p;
               p3 := p3 - 1;
               p3 := p3 - p3 : 35×35 + 1;
               even := p3 = p3 : 2×2;
               if p3 > 9 then
               p3 := if p3 < 19 then p3 + 39
               else
               if p3 < 28 then p3 + 14 else p3 - 10;
               if even = red then
               begin
                  skrvtegn(if red then 62 else 29);
                  red := -, red
               end;
               skrvtegn(p3)
            end for s;
            LINE
         end for r
      end inner MAP-block;
      LINE;  LINE; LINE;
      SHIFT(4);
      skrvtegn(62);
      skrvtekst(ǂ<Hvis De ønsker flere kort, saa skriv et 1-tal her: ǂ);
      r := tasttegn;
      go to if r = 1 then LL else RESTART
   end of MAP-block;
LINEQ:
   begin comment LINEQ-block;
      integer N, i, j;
      SHIFT(10);
      skrvtekst(ǂ<Type 3.  Løsning af tilfældige lineære ligninger.ǂ);
      LINE;  LINE;
      if newlineq then
      begin
         SHIFT(5);
         skrvtekst(ǂ<
     Programmet genererer  og løser et sæt af N tilfældige lineære ligninger.
Maximumværdien af N er 24.  Regnetiden for N = 20 er 23 sec.
i ALGOL og 4 sec. i maskinsprog.  Opgiv N = 0 for stop.
ǂ);
         linerest := linerest - 4;
         newlineq := false
      end if newlineq;
MM:   LINE;
      SHIFT(4);
      skrvtekst(ǂ<Opgiv N: ǂ);
      N := tast;
      LINE;
      if N > 24 then N := 24;
      if N = 0 then go to RESTART;
      begin comment inner LINEQ-block;
```

```
         array x[1:N], MATRIX[1:N, 1:N + 1];
         procedure  LINEQ1 (N, a, x, NOSOLUTION);
         integer  N;
         array  a, x;
         label  NOSOLUTION;
         begin
            integer p, i, j;
            real  M;
            for  p := 1 step 1 until N − 1 do
            begin
               for  i := p + 1 step 1 until N do
               begin
                  if  a[p,p] ≠ 0 then go to L2;
                  if  a[i,p] ≠ 0 then go to L1;
                  if  i < N then go to L3;
                  go to NOSOLUTION;
L1:               for j := p step 1 until N + 1 do
                  begin
                     M := a[p,j];
                     a[p,j] := a[i,j];
                     a[i,j] := M
                  end of row exchange;
                  go to L3;
L2:               if a[i,p] = 0 then go to L3;
                  M := −a[i,p]/a[p,p];
                  for j := p+1 step 1 until N+1 do
                  a[i,j] := a[i,j] + M×a[p,j];
L3:            end for i;
            end for p;
            if a[N,N] = 0 then go to NOSOLUTION;
            for p := N step −1 until 1 do
            begin
               x[p] := a[p,N+1] := a[p,N+1]/a[p,p];
               if p = 1 then go to L4;
               for i := p−1 step −1 until 1 do
               a[i,N+1] := a[i,N+1] − x[p]×a[i,p]
            end for second p;
L4:      end LINEQ−1;
NN:      for i := 1 step 1 until N do
         for j := 1 step 1 until N+1 do
         MATRIX[i,j] := RANDOM(30000);
         LINEQ1(N, MATRIX, x, ERROR);
         go to MM;
ERROR:   SHIFT(4);  LINE;
         skrvtekst(‹<Undskyld, determinanten er nul.  Her er et andet eksempel.›);
         LINE;
         go to NN
      end of inner LINEQ−block
   end of LINEQ−block;
PRIME:
   begin comment prime block;
      boolean first, last, small;
      integer type, num, num1, fact, count, A, B;
      integer procedure PRIM1(x);
      integer x;
      begin
         integer y;
A:       PRIM1 := x := x + 2;
         y := 1;
         for y := y + 2 while y×y≤x do
         if (x:y)×y = x then go to A
      end;
      procedure READ(number, text);
      integer number;
      string text;
      begin
         real N;
```

```
PP:      SHIFT(4);
         skrvtekst(text);
         N := tast;
         LINE;
         if N < 1 ∨ N > 536870911 then go to PP;
         number := N
      end READ;
      SHIFT(10);
      skrvtekst(≮<Type 4.  Beregning af primtal.≯);
      LINE;  LINE;
      if newprime then
      begin
         SHIFT(8);
         skrvtekst(≮<
     Programmet indeholder to beregningstyper:
     1.  Beregning af primfaktorer i et opgivet tal, N.
     2.  Beregning af primtal i et opgivet interval fra A til B.
     Opgiv type 3 for stop.  Øvre grænse for tallene er 536870911.
≯);
         linerest := linerest − 5;
         newprime := false
      end of newprime;
OO:   LINE;
      READ(type, ≮<Opgiv primtalberegningstype: ≯);
      if type = 1 then
      begin
         READ(num, ≮<Opgiv tallet, N: ≯);
         num1 := num;
         first := true;
         last := false;
         if num < 4 then
QQ:      begin
            skrvtekst(≮<Primtal≯);
            go to OO
         end;
         for fact := 2, 3, PRIM1(fact) while fact×fact ≤ num1 ∧ num > 1 do
         begin
            count := 0;
RR:         if num:fact×fact = num then
SS:         begin
               count := count + 1;
               num := num:fact;
               if first then skrvtekst(≮<=≯);
               if count = 1 then
               begin
                  if ¬, first then skrvtekst(≮<×≯);
                  first := false;
                  skrv(if fact < 10 then ≮d≯
                  else if fact < 100 then ≮dd≯
                  else if fact < 1000 then ≮ddd≯
                  else if fact < 10000 then ≮dddd≯
                  else ≮ddddddddd≯, fact)
               end if count = 1;
               go to if last then OO else RR
            end if divisor;
            if count > 1 then
            begin
               skrvtekst(≮<↑≯);
               skrv(if count < 10 then ≮d≯ else ≮dd≯, count)
            end if power printing;
            count := 0
         end for fact;
         if first then go to QQ;
         last := true;
         fact := num;
         if num > 1 then go to SS;
         go to OO
      end if type = 1
      else
      if type = 2 then
      begin
         READ(A, ≮<Opgiv nedre grænse, A: ≯);
         READ(B, ≮<Opgiv øvre grænse, B: ≯);
         small := B < 10000;
         count := 0;
         fact := if small then 10 else 8;
         SHIFT(4);
         if A = 1 then A := 2
         else
         if A > 3 then
         begin
            A := A − (if A:2×2 = A then 1 else 2);
UU:         A := PRIM1(A)
         end;
TT:      if count:fact×fact = count then LINE;
         count := count + 1;
         if A≤B then skrv(if small then ≮−ddddd≯ else ≮−dddddddd≯, A);
         if A < B then
         begin
            if A = 2 then
            begin
               A := 3;
               go to TT
            end;
            go to UU
         end;
         go to OO
      end type = 2
      else
      go to RESTART
   end prime block;
LANU:
   begin comment large number calculation block;
      boolean first, out;
      integer M, carry, count, c1, c2, d1, d2, type, N, alimit, asize, nn, a,
              b, D, bsize, m;
      procedure READ(number, text);
      integer number;
      string text;
      begin
         SHIFT(4);
         skrvtekst(text);
         number := tast;
         LINE
      end READ;
      procedure ALARM(n);
      value n;
      integer n;
      skrvtekst(≮<
FEJL ≯, if n = 1 then ≮<1≯ else ≮<2≯);
      procedure MULT(n, A, size);
      value n;
      integer n, size;
      integer array A;
      begin
         carry := 0;
         for count := 0 step 1 until alimit do
         begin
            c1 := A[count];
            c2 := c1:M;
            c1 := (c1−c2×M)×n + carry;
            carry := c1:M;
            c1 := c1 − carry×M;
            c2 := c2×n + carry;
            carry := c2:M;
            A[count] := (c2−carry×M)×M + c1;
            if count = size then
            begin
               if carry = 0 then go to ex
               else
               if count < alimit then size := size + 1
               else
               ALARM(1)
            end if count
         end for count;
ex:   end MULT;
      procedure DIV(n, A, size, empty);
      value n;
      integer n, size;
      boolean empty;
      integer array A;
      begin
         first := true;
         carry := 0;
         for count := size step −1 until 0 do
         begin
            c1 := A[count];
            c2 := c1:M;
            c1 := c1 − c2×M;
            carry := carry×M + c2;
            c2 := carry:n;
            carry := (carry − c2×n)×M + c1;
            c1 := carry:n;
            carry := carry − c1×n;
            A[count] := c1 := c1 + c2×M;
            if first then
            begin
               if c1 > 0 then first := false
               else
               if size > 0 then size := size − 1
            end if first
         end for count;
         empty := first ∧ c1 = 0
      end DIV;
      procedure ADD(plus, A, B, asize, bsize);
      value plus, bsize;
      boolean plus;
      integer asize, bsize;
      integer array A, B;
      begin
         carry := 0;
         for count := 0 step 1 until alimit do
         begin
            c1 := A[count];  d1 := B[count];
            c2 := c1:M;      d2 := d1:M;
            c1 := c1−c2×M;   d1 := d1 − d2×M;
            c1 := c1 + (if plus then d1 else − d1) + carry;
            carry := 0;
L1:         if c1 < 0 then
            begin
               c1 := c1 + M;
               carry := carry − 1;
               go to L1
            end if c1 negative;
            d1 := c1:M;
            c1 := c1 − d1×M;
            c2 := c2 + (if plus then d2 else − d2) + d1 + carry;
            carry := 0;
L2:         if c2 < 0 then
            begin
               c2 := c2 + M;
               carry := carry − 1;
               go to L2
            end if c2 negative;
            d1 := c2:M;
            c2 := c2 − d1×M;
            carry := carry + d1;
            A[count] := c1 := c1 + c2×M;
            if count ≥ bsize ∧ carry = 0 then go to L3
         end for count;
         if carry ≠ 0 then ALARM(2);
L3:      first := true;
         for count := alimit step −1 until 0 do
         begin
            asize := count;
            if A[count] ≠ 0 then go to L4
         end;
L4:   end ADD;
      procedure P4(n);
      value n;
      integer n;
      begin
         integer i, z, D, a;
         D := 1000;
         z := if first then 0 else 16;
         for i := 1 step 1 until 4 do
         begin
            a := n:D;
            n := n − a×D;
            if a ≠ 0 then
            begin
               skrvtegn(a);
               first := false;
               z := 16
            end
            else skrvtegn(z);
            D := D:10
         end for i
      end P4;
      procedure PR(A, size);
      value size;
      integer size;
      array A;
      begin
         first := true;
         d1 := 0;
         for count := size step −1 until 0 do
         begin
            c1 := A[count];
            c2 := c1:M;
            c1 := c1 − c2×M;
            P4(c2);
            skrvml(1);
            P4(c1);
            skrvml(1);
            d1 := d1 + 1;
            if d1:8×8 = d1 then LINE
         end for count
      end PR;
      SHIFT(10);
      skrvtekst(‹<Type 5.  Beregning af store tal.›);
      LINE;  LINE;
      if newlanu then
      begin
         SHIFT(8);
         skrvtekst(‹<
        Programmet indeholder fire beregningstyper:
        1. Beregning af fakultet: FAC(N) = 1×2×3×4........×N.
        2. Beregning af potens: a↑b.
        3. Beregning af e = 2.718 ...... med D cifre.
        4. Beregning af pi = 3.1415.... med D cifre.
        5. giver programstop.
≯);
         linerest := linerest − 7;
         newlanu := false
      end of newlanu;
      LINE;
      M := 10000;
VV:   READ(type, ≮<Opgiv beregningstype for store tal: ≯);
      if type = 1 then
      begin
         READ(N, ≮<Opgiv N:≯);
         if N > 1000 then N := 1000;
         alimit := 0.05×N×ln(N);
         begin
            integer array FAC[0:alimit];
            for count := 0 step 1 until alimit do FAC[count] := 0;
            asize := 0;
            FAC[0] := 1;
            for nn := 1 step 1 until N do MULT(nn, FAC, asize);
            LINE;  LINE;
            SHIFT(4);
            skrvtekst(≮<FAC :=≯);
            LINE;
            PR(FAC, asize);
            LINE;
            go to VV
         end block
      end if type = 1
      else
      if type = 2 then
      begin
         READ(a, ≮<Opgiv a: ≯);
         READ(b, ≮<Opgiv b: ≯);
         alimit := 1 + 0.055×b×ln(a);
         begin
            integer array POT[0:alimit];
            for count := 0 step 1 until alimit do POT[count] := 0;
            asize := 0;
            POT[0] := 1;
            for nn := 1 step 1 until b do MULT(a, POT, asize);
            LINE;  LINE;
            SHIFT(4);
            skrvtekst(≮<a↑b := ≯);
            LINE;
            PR(POT, asize);
            LINE;
            go to VV
         end block
      end if type = 2
      else
      if type < 5 then
      begin
         READ(D, ≮<Opgiv D: ≯);
         alimit := D:8;
         if alimit×8 ≠ D then
         begin
            skrvtekst(≮<D er ændret til:≯);
            alimit := alimit + 1;
            D := 8×alimit;
            skrv(≮nddd≯, D);
            LINE
         end;
         D := 8×alimit
      end
      else go to RESTART;
      if type = 3 then
      begin
         integer array RESULT, TERM[0:alimit];
         for count := 0 step 1 until alimit do
         RESULT[count] := TERM[count] := 0;
         asize := bsize := alimit;
         RESULT[alimit] := 2;
         TERM[alimit] := 1;
         out := false;
         m := 1;
         for m := m + 1 while −, out do
         begin
            DIV(m, TERM, bsize, out);
            ADD(true, RESULT, TERM, asize, bsize)
         end for m;
         LINE;  LINE;
         SHIFT(4);
         skrvtekst(‹<e×10↑D := ›);
         LINE;
         PR(RESULT, asize);
         LINE;
         go to VV
      end block if type = 3
      else
      begin
         boolean out1, out2, out3, plus;
         integer t1size, t2size, t3size, ssize;
         integer array RESULT, T1, T2, T3, SUM[0:alimit];
         for count := 0 step 1 until alimit do
         RESULT[count] := T1[count] := T2[count] := T3[count] := 0;
         T1[alimit] := T2[alimit] := T3[alimit] := 24;
         asize := t1size := t2size := t3size := alimit;
         DIV(8, T1, t1size, out1);
         DIV(171, T2, t2size, out2);
         DIV(1434, T3, t3size, out3);
         plus := false;
         m := −1;
         for m := m + 2 while −, out1 do
         begin
            for count := 0 step 1 until alimit do SUM[count] := 0;
            ssize := 0;
            ADD(true, SUM, T1, ssize, t1size);
            if −, out2 then ADD(true, SUM, T2, ssize, t2size);
            if −, out3 then ADD(true, SUM, T3, ssize, t3size);
            DIV(m, SUM, ssize, out);
            plus := −, plus;
            ADD(plus, RESULT, SUM, asize, ssize);
            DIV(64, T1, t1size, out1);
            if −, out2 then DIV(3249, T2, t2size, out2);
            if −, out3 then for nn := 1,2 do DIV(239, T3, t3size, out3)
         end for m;
         LINE;  LINE;
         SHIFT(4);
         skrvtekst(‹<pi×10↑D :=›);
         LINE;
         PR(RESULT, asize);
         LINE;
         go to VV
      end type = 4
   end LANU-block;
FINISH:
end of program DEMON-1D;
```