```
algol,i,n<
This program was made by my father, Jørgen Kjær,
while he worked for Haldor Topsøe.

This is Service Pack 1; a few bugs have been corrected:

1: Call of where moved out of show loop.
2: Error in shift code in LONGMULT.
3: Calculation of asize in ASSIGN has been changed.

ASSIGN and SQRT are now called after reading the number with read real.

2011-Jul-27 22.21 / TN: Experimenting with shifting, LONGMULT performance, and LONGDI

Timing (in seconds), 380 decimals, buffer GIER, no index check:

                Classic GA4     Turbo GA4       Save, pct.
sqrt(r):        5331.8          4923.3           7.7
sqrt2(r):       1247.6          1077.9          13.6
sqrt3(r):        389.2           365.0           6.2
sqrt(B):         388.1           363.9           6.2

Timing (in seconds), 380 decimals, no buffer GIER, no index check:

                Classic GA4     Turbo GA4       Save, pct.
sqrt(r):        8195.6          7892.0           3.7
sqrt2(r):       2147.2          2008.1           6.5
sqrt3(r):        379.9           364.9           3.9
sqrt(B):         377.6           361.5           4.3

Program DEMON-5. Calculation of large numbers.
begin
   boolean first, empty, show, large;
   boolean showAll;
   integer linerest, lang, decimals, limit, carry, count, MODUL, cell,
   cell2, asize, bsize, csize, type, TYPE, D, E, FREE, ftrack, step, c39;
   procedure NEW PAGE;
   begin
      for linerest := linerest - 1 while linerest ≥ 0, 69 do writecr;
      writechar(72)
   end NEW PAGE;
   procedure LINE;
   if linerest < 8 then NEW PAGE
   else
   begin
      comment linerest := linerest - 1;
      writecr
   end LINE;
   procedure WRITE TEXT(dan, eng, fr, ger);
   string dan, eng, fr, ger;
   writetext(case lang of (dan, eng, fr, ger));
   procedure SELECT LANGUAGE;
   begin
      LINE;
      writetext(
      ┤<Select language: d: danish, e: english, f: french, g: german.: ├);
      lang := lyn - 51;
      if lang < 1 then lang := 1;
      if lang > 4 then lang := 4;
      LINE;
      WRITE TEXT(
      ┤<Dansk├,
      ┤<English├,
      ┤<Francais├,
```

```
   ┤<Deutsch┤);
   LINE
end SELECT LANGUAGE;
integer procedure ASK NUMBER(dan, eng, fr, ger);
string dan, eng, fr, ger;
begin
   LINE;
   WRITE TEXT(dan, eng, fr, ger);
   writetext(┤<: ┤);
   ASK NUMBER := read integer
end ASK NUMBER;
procedure ACCEPT(cond);
value cond;
boolean cond;
if -, cond then
begin
   LINE;
   WRITE TEXT(
   ┤<Brug flere heltalscifre┤,
   ┤<Use more integer digits┤,
   ┤<Le nombre de chiffres entiers est trop petit┤,
   ┤<Zu wenig Ganzzahlstellen┤);
   go to if show then E1 else E2
end ACCEPT;
procedure ALARM(text);
string text;
begin
   LINE;
   writetext(┤<Error in: ┤);
   writetext(text);
   go to if show then E1 else E2
end ALARM;
integer stat TO REAL get A;
integer stat COMPARE get A;
integer stat COMPARE get B;
integer stat LONGMULT2 get A;
integer stat LONGMULT2 get B;
integer stat LONGMULT2 get C 1;
integer stat LONGMULT2 get C 2;
integer stat LONGMULT2 put A;
integer stat LONGMULT2 put B;
integer stat LONGMULT2 put C 1;
integer stat LONGMULT2 put C 2;
integer stat LONGMULT2 put C 3;
integer stat LONGDIVIDE2 get A 1;
integer stat LONGDIVIDE2 get A 2;
integer stat LONGDIVIDE2 get A 3;
integer stat LONGDIVIDE2 get A 4;
integer stat LONGDIVIDE2 get A 5;
integer stat LONGDIVIDE2 get B 1;
integer stat LONGDIVIDE2 get B 2;
integer stat LONGDIVIDE2 get B 3;
integer stat LONGDIVIDE2 put A 1;
integer stat LONGDIVIDE2 put A 2;
integer stat LONGDIVIDE2 put A 3;
integer stat LONGDIVIDE2 put A 4;
integer stat LONGDIVIDE2 put A 5;
integer stat LONGDIVIDE2 put C 1;
integer stat LONGDIVIDE2 put C 2;
integer stat LONGMULT get B 1;
integer stat LONGMULT get B 2;
integer stat LONGMULT get RES 1;
integer stat LONGMULT get RES 2;
integer stat LONGMULT put RES;
```

```
procedure STATISTICS PROCESS(p);
procedure p;
begin
   integer stat counter;
   p(stat counter, {<TO REAL get A }, stat TO REAL get A);
   p(stat counter, {<COMPARE get A }, stat COMPARE get A);
   p(stat counter, {<COMPARE get B }, stat COMPARE get B);
   p(stat counter, {<LONGMULT2 get A }, stat LONGMULT2 get A);
   p(stat counter, {<LONGMULT2 get B }, stat LONGMULT2 get B);
   p(stat counter, {<LONGMULT2 get C 1 }, stat LONGMULT2 get C 1);
   p(stat counter, {<LONGMULT2 get C 2 }, stat LONGMULT2 get C 2);
   p(stat counter, {<LONGMULT2 put A }, stat LONGMULT2 put A);
   p(stat counter, {<LONGMULT2 put B }, stat LONGMULT2 put B);
   p(stat counter, {<LONGMULT2 put C 1 }, stat LONGMULT2 put C 1);
   p(stat counter, {<LONGMULT2 put C 2 }, stat LONGMULT2 put C 2);
   p(stat counter, {<LONGMULT2 put C 3 }, stat LONGMULT2 put C 3);
   p(stat counter, {<LONGDIVIDE2 get A 1}, stat LONGDIVIDE2 get A 1);
   p(stat counter, {<LONGDIVIDE2 get A 2}, stat LONGDIVIDE2 get A 2);
   p(stat counter, {<LONGDIVIDE2 get A 3}, stat LONGDIVIDE2 get A 3);
   p(stat counter, {<LONGDIVIDE2 get A 4}, stat LONGDIVIDE2 get A 4);
   p(stat counter, {<LONGDIVIDE2 get A 5}, stat LONGDIVIDE2 get A 5);
   p(stat counter, {<LONGDIVIDE2 get B 1}, stat LONGDIVIDE2 get B 1);
   p(stat counter, {<LONGDIVIDE2 get B 2}, stat LONGDIVIDE2 get B 2);
   p(stat counter, {<LONGDIVIDE2 get B 3}, stat LONGDIVIDE2 get B 3);
   p(stat counter, {<LONGDIVIDE2 put A 1}, stat LONGDIVIDE2 put A 1);
   p(stat counter, {<LONGDIVIDE2 put A 2}, stat LONGDIVIDE2 put A 2);
   p(stat counter, {<LONGDIVIDE2 put A 3}, stat LONGDIVIDE2 put A 3);
   p(stat counter, {<LONGDIVIDE2 put A 4}, stat LONGDIVIDE2 put A 4);
   p(stat counter, {<LONGDIVIDE2 put A 5}, stat LONGDIVIDE2 put A 5);
   p(stat counter, {<LONGDIVIDE2 put C 1}, stat LONGDIVIDE2 put C 1);
   p(stat counter, {<LONGDIVIDE2 put C 2}, stat LONGDIVIDE2 put C 2);
   p(stat counter, {<LONGMULT get B 1 }, stat LONGMULT get B 1);
   p(stat counter, {<LONGMULT get B 2 }, stat LONGMULT get B 2);
   p(stat counter, {<LONGMULT get RES 1}, stat LONGMULT get RES 1);
   p(stat counter, {<LONGMULT get RES 2}, stat LONGMULT get RES 2);
   p(stat counter, {<LONGMULT put RES }, stat LONGMULT put RES);
end STATISTICS PROCESS;
procedure STATISTICS INIT;
begin
   procedure init( c, t, s );
   value c;
   integer c, s;
   string t;
   begin
      s := 0
   end init;
   STATISTICS PROCESS( init )
end STATISTICS INIT;
integer procedure STATISTICS PRINT;
begin
   procedure print( c, t, s );
   value c;
   integer c, s;
   string t;
   begin
      LINE;
      writetext( t );
      writetext( {<: } );
      writeinteger( {-dddddd}, s )
   end init;
   STATISTICS PROCESS( print )
end STATISTICS INIT;
integer procedure ASSIGN(x, A, asize, na);
value x, na;
```

```
   integer asize, na;
   real x;
   integer array A;
   begin
      integer c1, c2, t1, t2, cell1, cell2;
      x := abs(x);
      c1 := c39;
      for count := 0 step 1 until c1 do A[count] := 0;
      if x = 0 then
      begin
         asize := c1 := c2 := cell1 := cell2 := 0;
         go to L1
      end if x = 0;
      comment Normalize x so that 1₁₀10 > x ≥ 1;
      asize := 0;
      if x ≥ MODUL then
      begin
  for x := x while x ≥ MODUL do
  begin
     asize := asize+1;
     x := x/MODUL
  end
      end
      else if x < 1 then
      for x := x while x < 1 do
      begin
         asize := asize-1;
  x := x×MODUL
      end;
      if asize > limit then ALARM(‡<ASSIGN>);
      cell1 := entier(x);
      cell2 := (x - cell1)×MODUL;
      c1 := asize - decimals;
      c2 := c1 - 1;
      if c2 < 0 then
      begin
         c2 := c1;
         cell2 := cell1
      end if c2 < 0;
      if c1 < 0 then c1 := c2 := cell1 := cell2 := 0;
L1: if large then
      begin
         t1 := 1 + c1:40;
         t2 := 1 + c2:40;
         c1 := c1 mod 40;
         c2 := c2 mod 40;
         for count := 1 step 1 until step do
         begin
            if count = t1 then
            begin
               A[c1] := cell1;
               if t1 ≠ t2 then
               begin
                  put(A, FREE, na×step + t1);
                  A[c1] := 0;
                  A[c2] := cell2;
                  put(A, FREE, na×step + t2)
               end different track
               else
               begin
                  A[c2] := cell2;
                  put(A, FREE, na×step + t1)
               end same track;
               A[c1] := A[c2] := 0
```

```
            end this track
            else
            put(A, FREE, na×step + count)
         end for count
      end if large
      else
      begin
         A[c1] := cell1;
         A[c2] := cell2
      end core
   end ASSIGN;
   integer procedure MULT(A, asize, na, n);
   value na, n;
   integer asize, na, n;
   integer array A;
   begin
      integer c, ta, c1, asize0;
      asize0 := asize;
      carry := c := 0;
      ta := na×step + 1;
      if large then get(A, FREE, ta);
      c1 := limit - decimals;
      for count := 0 step 1 until c1 do
      begin
         cell := if count > asize0 - decimals then 0 else A[c];
         code cell, MODUL, carry, n;
         2, 44;
         2, 44;
         2, 44;
         3, 44;
         arn a3, pm a1 ; R := carry, M := cell
         ml p+a4,dl a2 ; RM := (carry+cell×n)/MODUL
         gr a3, gm a1 ; carry := quotient, cell := rem.
         e ;
         A[c] := cell;
         c := c + 1;
         if large then
         begin
            if c = 40 then
            begin
               c := 0;
               put(A, FREE, ta);
               ta := ta + 1;
               get(A, FREE, ta)
            end if c = 40
         end if large;
         if count = asize - decimals then
         begin
            if carry = 0 then go to EX
            else
            if count < c1 then asize := asize + 1
            else ALARM(‡<MULT‡)
         end if asize
      end for count;
EX: if large then put(A, FREE, ta)
   end MULT;
   integer procedure DIVIDE(A, asize, na, n, empty);
   value na, n;
   integer asize, na, n;
   boolean empty;
   integer array A;
   begin
      integer c, ta;
      first := true;
```

```
   carry := 0;
   c := asize - decimals;
   ta := 1 + c:40 + na×step;
   c := c mod 40;
   if large then get(A, FREE, ta);
   for count := asize step -1 until decimals do
   begin
      cell := A[c];
      code cell, MODUL, carry, n;
      2, 44;
      2, 44;
      2, 44;
      3, 44;
      arn a1, pm a3 ; R := cell, M := carry
      ml a2, dl p+a4; RM := (cell+carry×MODUL)/n
      gr a1, gm a3 ; cell := quotient, carry := rem.
      e ;
      A[c] := cell;
      c := c - 1;
      if large then
      begin
         if c < 0 then
         begin
            c := 39;
            put(A, FREE, ta);
            ta := ta - 1;
            get(A, FREE, ta)
         end if c < 0
      end if large;
      if first then
      begin
         if cell > 0 then first := false
         else
         if asize > decimals then asize := asize - 1
      end if first
   end for count;
   if large then put(A, FREE, ta);
   empty := first ∧ cell = 0
end DIVIDE;
integer procedure PRINT(A, asize, na);
value asize, na;
integer asize, na;
integer array A;
begin
   boolean first;
   integer DIVISOR, digit, i, space, group, ta, c;
   integer asize0;
   procedure GROUP(n);
   value n;
   integer n;
   begin
      DIVISOR := MODUL:10;
      space := if first then 0 else 16;
      for i := 1 step 1 until 10 do
      begin
         digit := n:DIVISOR;
         n := n mod DIVISOR;
         if digit ≠ 0 then
         begin
            writechar(digit);
            first := false;
            space := 16
         end
         else writechar(space);
```

```
         if i = 5 then writechar(0);
         DIVISOR := DIVISOR:10
      end for i
   end GROUP;
   if kbon ∨ true then
   begin
      LINE;
      writetext( ⟨<asize: ⟩ );
      writeinteger( ⟨-d⟩, asize )
   end;
   first := true;
   group := 0;
   LINE;
   comment if asize < 0 then asize := 0;
   asize0 := if asize < 0 then 0 else asize;
   c := asize0 - decimals;
   ta := 1 + c:40;
   c := c mod 40;
   if large then get(A, FREE, na×step + ta);
   for count := asize0 step -1 until decimals do
   begin
      GROUP(if count ≤ asize then A[c] else 0);
      if count = 0 ∧ decimals < 0 then
      begin
         writechar(59);
         first := false
      end
      else writechar(0);
      group := group + 1;
      if (group mod 6 = 0) ∧ count ≠ decimals then LINE;
      c := c - 1;
      if large then
      begin
         if c < 0 then
         begin
            c := 39;
            ta := ta - 1;
            get(A, FREE, na×step + ta)
         end if c < 0
      end if large
   end for count
end PRINT;
integer procedure COPY(A, asize, na, B, bsize, nb);
value na, nb;
integer asize, na, bsize, nb;
integer array A, B;
begin
   integer c, c1, t1, t2;
   c1 := c39;
   if large then
   begin
      t1 := na×step;
      t2 := nb×step;
      for count := 1 step 1 until step do
      begin
         t1 := t1 + 1;
         t2 := t2 + 1;
         get(A, FREE, t1);
         put(A, FREE, t2)
      end for count
   end if large
   else
   for c := 0 step 1 until c1 do B[c] := A[c];
   bsize := asize
```

```
   end COPY;
   integer procedure ADD(B, bsize, nb, factor, A, asize, na);
   value bsize, nb, factor, na;
   integer bsize, nb, factor, asize, na;
   integer array A, B;
   begin
      integer ta, tb, c;
      if large then
      begin
         ta := tb := 1;
         get(A, FREE, na×step + ta);
         get(B, FREE, nb×step + tb)
      end if large;
      c := - 1;
      carry := 0;
      for count := decimals step 1 until limit do
      begin
         c := c + 1;
         if c = 40 then
         begin
            c := 0;
            put(A, FREE, na×step + ta);
            ta := tb := ta + 1;
            get(A, FREE, na×step + ta);
            get(B, FREE, nb×step + tb)
         end if c = 40;
         comment cell := A[c] + factor×B[c] + carry;
         cell := (if count ≤ asize then A[c] else 0) + (if count ≤ bsize then factor×
         carry := 1;
         for carry := carry -1 while cell < 0 do
         cell := cell + MODUL;
         cell2 := cell:MODUL;
         A[c] := cell - cell2×MODUL;
         carry := carry + cell2;
         if count ≥ bsize ∧ carry = 0 then go to L1
      end for count;
L1: if carry ≠ 0 then ALARM(‹<ADD›);
      if large then put(A, FREE, na×step + ta);
      asize := limit + 1;
      c := limit - decimals;
      ta := 1 + c:40;
      c := c mod 40;
      if large then get(A, FREE, na×step + ta);
      for asize := asize -1 while asize > decimals do
      begin
         if A[c] ≠ 0 then go to L2;
         c := c - 1;
         if c < 0 then
         begin
            c := 39;
            ta := ta - 1;
            get(A, FREE, na×step + ta)
         end if c < 0
      end for asize;
L2:end ADD;
   real procedure TO REAL( A, asize, na );
   value asize, na;
   integer asize, na;
   integer array A;
   begin
      integer xa, ca, sa;
      real r, r0;
      r := 0.0;
      sa := -1;
```

```
   for xa := asize step -1 until decimals do
   begin
      begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa then begin sa
      r0 := r + A[ca] × MODUL ↑ xa;
      if r ≠ 0.0 ∧ r = r0 then goto TO REAL LOOP END;
      r := r0
   end;
   TO REAL LOOP END:
   TO REAL := r
end TO REAL;
integer procedure COMPARE(A, asize, na, B, bsize, nb, size, acell, bcell );
comment Compare A and B. Return:
      size: The index of the most significant differing cell
      acell and bcell: The actual differing cells
   If identical, size = decimals-1 and cells are zero;
value asize, na, bsize, nb;
integer asize, na, bsize, nb, size, acell, bcell;
integer array A, B;
begin
   integer xa, ca, sa, xb, cb, sb;
   sa := sb := -1;
   for size := if asize > bsize then asize else bsize step -1 until decimals do
   begin
      xa := size;
      begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa then begin sa
      acell := if xa > asize then 0 else A[ca];
      xb := size;
      begin cb := xb - decimals; if large then begin if cb : 40 ≠ sb then begin sb
      bcell := if xb > bsize then 0 else B[cb];
      if acell ≠ bcell then
      begin
         goto COMPARE LOOP END
      end
   end;
   size := decimals - 1;
   acell := bcell := 0;
   COMPARE LOOP END:
   if kbon ∧ false then
   begin
      LINE;
      writetext( ⊰<COMPARE: size = ⊱ );
      writeinteger( ⊰-d⊱, size );
      writetext( ⊰<, acell = ⊱ );
      writeinteger( ⊰-d⊱, acell );
      writetext( ⊰<, bcell = ⊱ );
      writeinteger( ⊰-d⊱, bcell );
      writetext( ⊰<, A = ⊱ );
      PRINT( A, asize, na );
      LINE;
      writetext( ⊰< B = ⊱ );
      PRINT( B, bsize, nb )
   end
end COMPARE;
integer procedure LONGMULT2(A, asize, na, B, bsize, nb, C, csize, nc);
value asize, na, bsize, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   integer xa, ca, sa, acell, xb, cb, sb, bcell, xc, cc, sc, ccell, xcmin;
   csize := decimals - 1; comment C := 0;
   sa := -1; comment No data in A buffer;
   sb := -1;
   sc := -1;
   for xb := decimals step 1 until bsize do
```

```
begin
   begin cb := xb - decimals; if large then begin if cb : 40 ≠ sb then begin sb
   bcell := B[cb];
   comment Ready to multiply A by bcell;
   carry := 0;
   if kbon ∧ false then
   begin
      LINE;
      writetext( ╞<B[╡ );
      writeinteger( ╞-d╡, xb );
      writetext( ╞<] = ╡ );
      writeinteger( ╞-d╡, bcell );
   end;
   for xa := if xb < 0 then decimals - xb - 1 else decimals step 1 until asize
   begin
      if xa > asize ∨ xa < decimals then
      begin
         acell := 0
      end
      else
      begin
         begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa then be
         acell := A[ca]
      end;
      xc := xa + xb;
      if xc > csize ∨ xc < decimals then
      begin
         xcmin := csize + 1;
         ccell := 0;
      end
      else
      begin
         xcmin := xc;
         begin cc := xc - decimals; if large then begin if cc : 40 ≠ sc then be
         ccell := C[cc]
      end;
      if kbon ∧ false then
      begin
         LINE;
         writetext( ╞< A[╡ );
         writeinteger( ╞-d╡, xa );
         writetext( ╞<] = ╡ );
         writeinteger( ╞-d╡, acell );
         writetext( ╞<, C[╡ );
         writeinteger( ╞-d╡, xc );
         writetext( ╞<] = ╡ );
         writeinteger( ╞-d╡, ccell );
      end;
      code acell, bcell, carry, ccell, MODUL;
      3, 44;
      3, 44;
      2, 44;
      3, 44;
      2, 44;
      arn a3 , ar p+a4 ; R := carry + ccell;
      pm p+a1, ml p+a2 ; RM := acell×bcell + carry + ccell;
      dl a5 , gr a3 ; RM := RM/MODUL; carry := quotient;
      gm p+a4 ; ccell := remainder
      e ;
      if (ccell ≠ 0 ∨ csize ≥ xc) ∧ xc ≥ decimals then
      begin
         if xc > csize then
         begin
            csize := xc
```

```
            end;
            for xc := xcmin step 1 until xa + xb do
            begin
               begin cc := xc - decimals; if large then begin if cc : 40 ≠ sc then
               C[cc] := if xc < xa + xb then 0 else ccell;
               if kbon ∧ false then
               begin
                  LINE;
                  writetext( ⎨< C[⎬ );
                  writeinteger( ⎨-d⎬, xc );
                  writetext( ⎨</⎬ );
                  writeinteger( ⎨-d⎬, cc );
                  writetext( ⎨<] = ⎬ );
                  writeinteger( ⎨-d⎬, C[cc] );
               end
            end
         end
      end;
      if carry ≠ 0 then
      begin
         ALARM(⎨<LONGMULT2⎬)
      end
   end;
   if large then begin if sc ≥ 0 then begin put( C, FREE, nc×step + 1 + sc ); stat
end LONGMULT2;
integer procedure LONGDIVIDE(A, asize, na, B, bsize, nb, C, csize, nc);
comment ( C, A ) := ( A : B, A mod B );
value na, bsize, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   LONGDIVIDE2(A, asize, na, B, bsize, nb, C, csize, nc, decimals);
end LONGDIVIDE;
integer procedure LONGDIVIDE2(A, asize, na, B, bsize, nb, C, csize, nc, decs);
comment ( C, A ) := ( A : B, A mod B );
value na, bsize, nb, nc, decs;
integer asize, na, bsize, nb, csize, nc, decs;
integer array A, B, C;
begin
   integer xa, ca, sa, xb, cb, sb, xc, cc, sc, bn, bn2, acell, bcell;
   integer an1, an, q, q0, digit, carry2, xamin, asize2, normfactor;
   if bsize < decs then
   begin
      ALARM(⎨<LONGDIVIDE2 1⎬)
   end;
   normfactor := 1;
   NORMALIZE LOOP START:
      xb := bsize;
      sb := -1;
      begin cb := xb - decimals; if large then begin if cb : 40 ≠ sb then begin sb
      bn := B[cb];
      if bn = 0 then
      begin
         ALARM(⎨<LONGDIVIDE2 2⎬)
      end;
      if kbon then
      begin
         LINE;
         writetext( ⎨<LONG DIVIDE: A:⎬ );
         PRINT( A, asize, na );
         LINE;
         writetext( ⎨< B:⎬ );
         PRINT( B, bsize, nb );
         LINE;
```

```
      writetext( ǂ< bn = ǂ );
      writeinteger( ǂ-dǂ, bn )
   end;
   if bn ≥ MODUL : 2 then goto NORMALIZE LOOP END;
   normfactor := MODUL : (bn + 1);
   if kbon then
   begin
      LINE;
      writetext( ǂ<LONG DIVIDE: normfactor = ǂ );
      writeinteger( ǂ-dǂ, normfactor )
   end;
   MULT( A, asize, na, normfactor );
   MULT( B, bsize, nb, normfactor );
   goto NORMALIZE LOOP START;
NORMALIZE LOOP END:
bn2 := bn + 2;
sa := sc := -1;
csize := decs - 1;
for xc := asize - bsize step -1 until decs do
begin
   if kbon then
   begin
      LINE;
      writetext( ǂ<LONG DIVIDE: xc = ǂ );
      writeinteger( ǂ-dǂ, xc )
   end;
   q := 0;
   comment Outline of the loop between QLOOPSTART and QLOOPEND:
      while A[xa..] ≥ B[xb..] do
        q0 := guess at A[xa..]/B[xb..] which is not too large
        A[xa..] := A[xa..] - q0×B[xb..]
        q := q + q0;
   QLOOPSTART:
      xa := xc + bsize + 1;
      if xa > asize then
      begin
         an1 := 0
      end
      else
      begin
         begin ca := xa - decimals; if large then begin if ca : 40 ǂ sa then be
         an1 := A[ca]
      end;
      if an1 > 0 then goto QMORE;
      if xc + bsize < decs then goto QLOOPEND;
      for xa := xc + bsize step -1 until decs do
      begin
         if xa > asize then
         begin
            acell := 0
         end
         else
         begin
            begin ca := xa - decimals; if large then begin if ca : 40 ǂ sa then
            acell := A[ca]
         end;
         xb := xa - xc;
         if xb < decs then
         begin
            bcell := 0
         end
         else
         begin
            begin cb := xb - decimals; if large then begin if cb : 40 ǂ sb then
```

```
      bcell := B[cb]
   end;
   if acell > bcell then goto QMORE;
   if acell < bcell then goto QLOOPEND;
end;
QMORE:
xa := xc + bsize;
if xa > asize ∨ xa < decs then
begin
   an := 0
end
else
begin
   begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa then be
   an := A[ca]
end;
code an1, an, bn2, q0, MODUL;
3, 44;
3, 44;
3, 44;
3, 44;
2, 44;
; arn p+a2, ar p+a3 ; R := an + bn - 1;
; sr c42 , pm p+a1 ;
arn p+a2, pm p+a1 ; R := an; M := an1;
ml a5 , dl p+a3 ; q0 := (an1×MODUL + an) : bn2;
gr p+a4 ;
e ;
comment q0 := q0 - 2;
if kbon then
begin
   LINE;
   writetext( ⊧< QMORE: (⊧ );
   writeinteger( ⊧-d⊧, an1 );
   writetext( ⊧<,⊧ );
   writeinteger( ⊧-d⊧, an );
   writetext( ⊧<, ... ) : ⊧ );
   writeinteger( ⊧-d⊧, bn );
   writetext( ⊧< ... estimate: ⊧ );
   writeinteger( ⊧-d⊧, q0 );
end;
if q0 = 0 then
begin
   q0 := 1;
   if kbon then
   begin
      writetext( ⊧<, increased to ⊧ );
      writeinteger( ⊧-d⊧, q0 );
   end
end;
if q0 < 0 then
begin
   ALARM( ⊧<LONG DIVIDE 3⊧ )
end;
if q0 ≥ MODUL then
begin
   q0 := MODUL - 1;
   if kbon then
   begin
      writetext( ⊧<, reduced to ⊧ );
      writeinteger( ⊧-d⊧, q0 );
   end
end;
carry := digit := carry2 := 0;
```

```
asize2 := decs - 1;
for xb := if xc < 0 then decs - xc - 1 else decs step 1 until bsize + 1 d
begin
   if xb > bsize then
   begin
      bcell := 0
   end
   else
   begin
      begin cb := xb - decimals; if large then begin if cb : 40 ≠ sb then
      bcell := B[cb]
   end;
   if kbon ∧ false then
   begin
      LINE;
      writetext( ┤< (├ );
      writeinteger( ┤-d├, carry );
      writetext( ┤<,├ );
      writeinteger( ┤-d├, bcell );
      writetext( ┤<) × ├ );
      writeinteger( ┤-d├, q0 );
   end;
   code bcell, q0, carry, digit, MODUL;
   3, 44;
   3, 44;
   2, 44;
   3, 44;
   2, 44;
   arn a3 , pm p+a1 ; R := carry; M := bcell;
   ml p+a2, dl a5 ; (carry,digit) :=
   gr a3 , gm p+a4 ; (bcell×q0 + carry) : / mod MODUL;
   e ;
   if kbon ∧ false then
   begin
      writetext( ┤< -> (├ );
      writeinteger( ┤-d├, carry );
      writetext( ┤<,├ );
      writeinteger( ┤-d├, digit );
      writetext( ┤<)├ );
   end;
   xa := xc + xb;
   if xa > asize ∨ xa < decs then
   begin
      xamin := asize + 1;
      acell := 0
   end
   else
   begin
      xamin := xa;
      begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa then
      acell := A[ca]
   end;
   acell := acell - digit + carry2 + MODUL;
   carry2 := acell : MODUL - 1;
   acell := acell mod MODUL;
   if acell ≠ 0 then
   begin
      asize2 := xa
   end;
   if (acell ≠ 0 ∨ asize ≥ xa) ∧ xa ≥ decs then
   begin
      if xa > asize then
      begin
         ALARM( ┤<LONG DIVIDE 4├ )
```

```
            end;
            for xa := xamin step 1 until xc + xb do
            begin
               begin ca := xa - decimals; if large then begin if ca : 40 ≠ sa t
               A[ca] := if xa < xc + xb then 0 else acell;
               if kbon ∧ false then
               begin
                  LINE;
                  writetext( ⊱< A[⊰ );
                  writeinteger( ⊱-d⊰, xa );
                  writetext( ⊱<] := ⊰ );
                  writeinteger( ⊱-d⊰, A[ca] );
               end
            end
         end
      end;
      if carry ≠ 0 then
      begin
         ALARM( ⊱<LONG DIVIDE 5⊰ )
      end;
      if carry2 ≠ 0 then
      begin
         ALARM( ⊱<LONG DIVIDE 6⊰ )
      end;
      q := q + q0;
      if kbon then
      begin
         LINE;
         writetext( ⊱< q += ⊰ );
         writeinteger( ⊱-d⊰, q0 );
         writetext( ⊱< -> ⊰ );
         writeinteger( ⊱-d⊰, q );
         LINE;
         writetext( ⊱< asize ⊰ );
         writeinteger( ⊱-d⊰, asize );
         writetext( ⊱< -> ⊰ );
         writeinteger( ⊱-d⊰, asize2 );
      end;
      asize := asize2;
      goto QLOOPSTART;
QLOOPEND:
if q ≠ 0 ∧ csize < xc then
begin
   csize := xc
end;
if csize ≥ xc then
begin
   cc := xc - decimals;
   if large then
   begin
      if cc : 40 ≠ sc then
      begin
         if sc ≥ 0 then begin put( C, FREE, nc×step + 1 + sc ); stat LONGDIV
         sc := cc : 40
      end;
      cc := cc mod 40
   end;
   C[cc] := q;
   if kbon then
   begin
      LINE;
      writetext( ⊱< C[⊰ );
      writeinteger( ⊱-d⊰, xc );
      writetext( ⊱<] := ⊰ );
```

```
            writeinteger( ǂ-dǂ, q );
         end;
      end
   end;
   if large then begin if sc ≥ 0 then begin put( C, FREE, nc×step + 1 + sc ); stat
   if normfactor ǂ 1 then
   begin
      DIVIDE( A, asize, na, normfactor, empty );
      DIVIDE( B, bsize, nb, normfactor, empty )
   end
end LONGDIVIDE2;
integer procedure LONGMULT(A, asize, na, B, bsize, nb, C, csize, nc);
value asize, na, bsize, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   integer c, factor, rsize, nr, s, shift, cb, tb, c1, t1, c2, t2, s1,
   s2, s3;
   integer array RES[0:c39];
   nr := ftrack;
   ftrack := ftrack + 1;
   if kbon then
   begin
      LINE;
      writetext( ǂ<LONGMULT: A:ǂ );
      PRINT( A, asize, na );
      LINE;
      writetext( ǂ< B:ǂ );
      PRINT( B, bsize, nb );
   end kbon;
   ASSIGN(0, C, csize, nc);
   tb := 1 + nb×step;
   cb := 0;
   if large then
   begin
      get(B, FREE, tb);
      stat LONGMULT get B 1 := stat LONGMULT get B 1 + 1
   end;
   for c := 0 step 1 until bsize - decimals do
   begin
      shift := c + decimals;
      if large then
      begin
         factor := B[cb];
         cb := cb + 1;
         if cb = 40 then
         begin
            cb := 0;
            tb := tb + 1;
            get(B, FREE, tb);
            stat LONGMULT get B 2 := stat LONGMULT get B 2 + 1
         end new B track
      end large
      else
      factor := B[c];
      COPY(A, asize, na, RES, rsize, nr);
      MULT(RES, rsize, nr, factor);
      if shift ǂ 0 then
      begin
         s1 := if shift < 0 then -c else limit - decimals;
         s2 := - sign(shift);
         comment MK: Next line changed from limit - decimals - c;
         s3 := if shift < 0 then limit - decimals else - c;
         for s := s1 step s2 until s3 do
```

```
          begin
             if s < 0 ∨ s > limit - decimals then
             cell := 0
             else
             if large then
             begin
                t1 := nr×step + 1 + s:40;
                c1 := s mod 40;
                get(RES, FREE, t1);
                stat LONGMULT get RES 1 := stat LONGMULT get RES 1 + 1;
                cell := RES[c1]
             end large
             else
             cell := RES[s];
             c2 := s + shift;
             if c2 > limit - decimals then
             begin
               if cell ≠ 0 then ALARM(↯<LONGMULT↯)
             end if too big
             else
             if c2 ≥ 0 then
             begin
                if large then
                begin
                   t2 := nr×step + 1 + c2:40;
                   c2 := c2 mod 40;
                   get(RES, FREE, t2);
                   stat LONGMULT get RES 2 := stat LONGMULT get RES 2 + 1;
                   RES[c2] := cell;
                   put(RES, FREE, t2);
                   stat LONGMULT put RES := stat LONGMULT put RES + 1
                end if large
                else
                RES[c2] := cell
             end if not c2 > limit - decimals
          end for s
       end if shift ≠ 0;
       rsize := rsize + shift;
       ADD(RES, rsize, nr, 1, C, csize, nc)
    end for c;
    ftrack := ftrack - 1
end LONGMULT;
integer procedure EXP(X, xsize, nx, A, asize, na, XN, xnsize, nxn);
value xsize, nx, na, nxn;
integer xsize, nx, asize, na, xnsize, nxn;
integer array X, A, XN;
begin
    boolean out;
    integer tsize, nt, m;
    integer array TERM[0:c39];
    nt := ftrack;
    ftrack := ftrack + 1;
    ASSIGN(1, A, asize, na);
    COPY(X, xsize, nx, TERM, tsize, nt);
    ADD(X, xsize, nx, 1, A, asize, na);
    out := false;
    m := 1;
    for m := m + 1 while -, out do
    begin
       LONGMULT(X, xsize, nx, TERM, tsize, nt, XN, xnsize, nxn);
       COPY(XN, xnsize, nxn, TERM, tsize, nt);
       DIVIDE(TERM, tsize, nt, m, out);
       ADD(TERM, tsize, nt, 1, A, asize, na)
    end for m;
```

```
   ftrack := ftrack - 1
end EXP;
integer procedure PI TO(A, asize, na, T2, t2size, n2, T3, t3size, n3);
value na, n2, n3;
integer asize, na, t2size, n2, t3size, n3;
integer array A, T2, T3;
begin
   boolean out1, out2, out3, out;
   integer factor, m, ns, n1, ssize, t1size;
   integer array SUM, T1[0:c39];
   ns := ftrack;
   n1 := ns + 1;
   ftrack := ftrack + 2;
   ASSIGN(0, A, asize, na);
   ASSIGN(3, T1, t1size, n1);
   out1 := false;
   ASSIGN(24, T2, t2size, n2);
   DIVIDE(T2, t2size, n2, 171, out2);
   ASSIGN(24, T3, t3size, n3);
   DIVIDE(T3, t3size, n3, 1434, out3);
   factor := m := - 1;
   for m := m + 2 while -, out1 do
   begin
      ASSIGN(0, SUM, ssize, ns);
      ADD(T1, t1size, n1, 1, SUM, ssize, ns);
      if -, out2 then
      ADD(T2, t2size, n2, 1, SUM, ssize, ns);
      if -, out3 then
      ADD(T3, t3size, n3, 1, SUM, ssize, ns);
      DIVIDE(SUM, ssize, ns, m, out);
      factor := - factor;
      ADD(SUM, ssize, ns, factor, A, asize, na);
      DIVIDE(T1, t1size, n1, 64, out1);
      if -, out2 then
      DIVIDE(T2, t2size, n2, 3249, out2);
      if -, out3 then
      DIVIDE(T3, t3size, n3, 57121, out3)
   end for m;
   ftrack := ftrack - 2
end PI TO;
integer procedure SQRT(x, A, asize, na, B, bsize, nb, C, csize, nc);
value x, na, nb, nc;
real x;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   boolean empty;
   integer xsize, zsize, nx, nz, i, imax;
   integer array X, Z[0:c39];
   nx := ftrack;
   nz := nx + 1;
   ftrack := ftrack + 2;
   ASSIGN(x, X, xsize, nx);
   ASSIGN(sqrt(x), A, asize, na);
   ASSIGN(1/sqrt(x), Z, zsize, nz);
   imax := if asize > zsize then asize else zsize;
   imax := imax - decimals + 1;
   for i := 1 step 1 until imax do
   begin
      if kbon then
      begin
         LINE;
         writetext( ⟨<SQRT: i = ⟩ );
         writeinteger( ⟨p⟩, i );
```

```
          writetext( ⟨<, A =⟩ );
          PRINT( A, asize, na )
       end;
       LONGMULT(A, asize, na, Z, zsize, nz, B, bsize, nb);
       if kbon then
       begin
          LINE;
          writetext( ⟨< B =⟩ );
          PRINT( B, bsize, nb )
       end;
       ASSIGN(2, C, csize, nc);
       ADD(B, bsize, nb, -1, C, csize, nc);
       LONGMULT(Z, zsize, nz, C, csize, nc, B, bsize, nb);
       LONGMULT(B, bsize, nb, X, xsize, nx, C, csize, nc);
       COPY(B, bsize, nb, Z, zsize, nz);
       ADD(C, csize, nc, 1, A, asize, na);
       DIVIDE(A, asize, na, 2, empty);
if kbon ∧ false then
begin
          LINE;
          writetext(⟨<i: ⟩);
          write integer(⟨-ddddd⟩, i);
          writetext(⟨<, A: ⟩);
          PRINT(A, asize, na)
end
    end for i;
    if kbon then
    begin
       LINE;
       writetext( ⟨<SQRT end A = ⟩ );
       PRINT( A, asize, na )
    end;
    ftrack := ftrack - 2
 end SQRT;
 integer procedure SQUARE(A, asize, na, B, bsize, nb);
 comment B := A×A;
 value na, nb;
 integer asize, na, bsize, nb;
 integer array A, B;
 begin
    integer xsize, nx;
    integer array X[0:c39];
    nx := ftrack;
    ftrack := ftrack + 1;
    COPY( A, asize, na, X, xsize, nx );
    LONGMULT2( A, asize, na, X, xsize, nx, B, bsize, nb );
    ftrack := ftrack - 1
 end SQUARE;
 integer procedure SQRT2(x, A, asize, na, B, bsize, nb, C, csize, nc);
 value x, na, nb, nc;
 real x;
 integer asize, na, bsize, nb, csize, nc;
 integer array A, B, C;
 begin
    boolean empty;
    integer xsize, zsize, nx, nz, i, imax;
    integer array X, Z[0:c39];
    nx := ftrack;
    nz := nx + 1;
    ftrack := ftrack + 2;
    ASSIGN(x, X, xsize, nx);
    ASSIGN(sqrt(x), A, asize, na);
    ASSIGN(1/sqrt(x), Z, zsize, nz);
    imax := if asize > zsize then asize else zsize;
```

```
    imax := imax - decimals + 1;
    for i := 1 step 1 until imax do
    begin
       if kbon then
       begin
          LINE;
          writetext( {<SQRT2: i = } );
          writeinteger( {p}, i );
          writetext( {<, A =} );
          PRINT( A, asize, na )
       end;
       LONGMULT2(A, asize, na, Z, zsize, nz, B, bsize, nb);
       if kbon then
       begin
          LINE;
          writetext( {< B =} );
          PRINT( B, bsize, nb )
       end;
       ASSIGN(2, C, csize, nc);
       ADD(B, bsize, nb, -1, C, csize, nc);
       LONGMULT2(Z, zsize, nz, C, csize, nc, B, bsize, nb);
       LONGMULT2(B, bsize, nb, X, xsize, nx, C, csize, nc);
       COPY(B, bsize, nb, Z, zsize, nz);
       ADD(C, csize, nc, 1, A, asize, na);
       DIVIDE(A, asize, na, 2, empty);
if kbon ∧ false then
begin
          LINE;
          writetext({<i: });
          write integer({-ddddd}, i);
          writetext({<, A: });
          PRINT(A, asize, na)
end
    end for i;
    if kbon then
    begin
       LINE;
       writetext( {<SQRT2 end A = } );
       PRINT( A, asize, na )
    end;
    ftrack := ftrack - 2
 end SQRT2;
 integer procedure SQRT3(x, A, asize, na, B, bsize, nb, C, csize, nc);
 value x, na, nb, nc;
 real x;
 integer asize, na, bsize, nb, csize, nc;
 integer array A, B, C;
 begin
    ASSIGN( x, B, bsize, nb );
    SQRT4( A, asize, na, B, bsize, nb, C, csize, nc )
 end SQRT3;
 integer procedure SQRT4(A, asize, na, B, bsize, nb, C, csize, nc);
 comment A := sqrt(B);
 value na, nb, nc;
 integer asize, na, bsize, nb, csize, nc;
 integer array A, B, C;
 begin
    ASSIGN(sqrt(TO REAL( B, bsize, nb )), A, asize, na);
    SQRT5(A, asize, na, B, bsize, nb, C, csize, nc)
 end SQRT4;
 integer procedure SQRT5(A, asize, na, B, bsize, nb, C, csize, nc);
 comment A := sqrt(B) using A as starting value;
 value na, nb, nc;
 integer asize, na, bsize, nb, csize, nc;
```

```
integer array A, B, C;
begin
   integer xsize, nx, prevsize, size, acell, ccell;
   integer array X[0:c39];
   boolean empty;
   integer i, sd;
   boolean kbonSQRT5;
   kbonSQRT5 := false;
   nx := ftrack;
   ftrack := ftrack + 1;
   sd := 7;
   prevsize := decimals + 1;
   for i := 1, i + 1 while true do
   begin
      if kbon ∨ kbonSQRT5 then
      begin
         LINE;
         writetext( ⟊<SQRT5: i = ⟋ );
         writeinteger( ⟊p⟋, i );
         writetext( ⟊<, sd = ⟋ );
         writeinteger( ⟊p⟋, sd );
         writetext( ⟊<, A =⟋ );
         PRINT( A, asize, na )
      end;
      COPY( B, bsize, nb, X, xsize, nx );
      LONGDIVIDE( X, xsize, nx, A, asize, na, C, csize, nc );
      COMPARE( A, asize, na, C, csize, nc, size, acell, ccell );
      if kbon ∨ kbonSQRT5 then
      begin
         LINE;
         writetext( ⟊< prevsize = ⟋ );
         writeinteger( ⟊-d⟋, prevsize );
         writetext( ⟊<, size = ⟋ );
         writeinteger( ⟊-d⟋, size );
         writetext( ⟊<, acell = ⟋ );
         writeinteger( ⟊-d⟋, acell );
         writetext( ⟊<, ccell = ⟋ );
         writeinteger( ⟊-d⟋, ccell );
         writetext( ⟊<, C =⟋ );
         PRINT( C, csize, nc )
      end;
      if prevsize ≤ decimals ∨ size < decimals ∨ size = decimals ∧ abs (acell − cc
      begin
         goto SQRT5 LOOP END
      end;
      prevsize := size;
      ADD(C, csize, nc, 1, A, asize, na);
      DIVIDE(A, asize, na, 2, empty);
      sd := sd + sd
   end for i;
   SQRT5 LOOP END:
   if kbon then
   begin
      LINE;
      writetext( ⟊<SQRT5 end A = ⟋ );
      PRINT( A, asize, na )
   end;
   ftrack := ftrack − 1
end SQRT5;
integer procedure SQRT6(A, asize, na, B, bsize, nb, C, csize, nc);
comment A := sqrt(B) using A as starting value;
value na, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
```

```
begin
   integer xsize, nx, prevsize, size, acell, ccell;
   integer array X[0:c39];
   boolean empty;
   integer i, sd, decs;
   boolean kbonSQRT6;
   kbonSQRT6 := false;
   nx := ftrack;
   ftrack := ftrack + 1;
   sd := 7;
   prevsize := decimals + 1;
   decs := decimals;
   for i := 1, i + 1 while true do
   begin
      if kbon ∨ kbonSQRT6 then
      begin
         LINE;
         writetext( ⟨<SQRT6: i = ⟩ );
         writeinteger( ⟨p⟩, i );
         writetext( ⟨<, sd = ⟩ );
         writeinteger( ⟨p⟩, sd );
         writetext( ⟨<, decs = ⟩ );
         writeinteger( ⟨-d⟩, decs );
         writetext( ⟨<, A =⟩ );
         PRINT( A, asize, na )
      end;
      COPY( B, bsize, nb, X, xsize, nx );
      LONGDIVIDE2( X, xsize, nx, A, asize, na, C, csize, nc, if decs < decimals th
      COMPARE( A, asize, na, C, csize, nc, size, acell, ccell );
      if kbon ∨ kbonSQRT6 then
      begin
         LINE;
         writetext( ⟨< prevsize = ⟩ );
         writeinteger( ⟨-d⟩, prevsize );
         writetext( ⟨<, size = ⟩ );
         writeinteger( ⟨-d⟩, size );
         writetext( ⟨<, acell = ⟩ );
         writeinteger( ⟨-d⟩, acell );
         writetext( ⟨<, ccell = ⟩ );
         writeinteger( ⟨-d⟩, ccell );
         writetext( ⟨<, C =⟩ );
         PRINT( C, csize, nc )
      end;
      if prevsize ≤ decimals ∨ size < decimals ∨ size = decimals ∧ abs (acell - cc
      begin
         goto SQRT6 LOOP END
      end;
      prevsize := size;
      ADD(C, csize, nc, 1, A, asize, na);
      DIVIDE(A, asize, na, 2, empty);
      sd := sd + sd;
      decs := if size < 0 then 3×size else decimals
   end for i;
   SQRT6 LOOP END:
   if kbon then
   begin
      LINE;
      writetext( ⟨<SQRT6 end A = ⟩ );
      PRINT( A, asize, na )
   end;
   ftrack := ftrack - 1
end SQRT6;
integer procedure AGM( A, asize, na, B, bsize, nb, C, csize, nc );
comment (A,B) := agm(A,B) and C := sum( 2 ↑ (j+1) × C[j] ). See
```

```
      Eugene Salamin,  Computation of pi Using Arithmetic- Geometric Mean ,
      Math. Comp., vol. 30, no 135, July 1976, pp. 565-570;
value na, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   integer w1size, nw1, w2size, nw2, twoPower, i, prevw1size, size, acell, bcell,
   integer array W1, W2[0:c39];
   boolean kbonAGM;
   kbonAGM := false;
   nw1 := ftrack; ftrack := ftrack + 1;
   nw2 := ftrack; ftrack := ftrack + 1;
   ASSIGN( 0, C, csize, nc );
   twoPower := 1;
   prevw1size := decimals + 1;
   comment End loop by goto AGM LOOP END;
   for i := 1, i + 1 while true do
   begin
      COPY( A, asize, na, W1, w1size, nw1 );
      ADD( B, bsize, nb, -1, W1, w1size, nw1 );
      COMPARE( A, asize, na, B, bsize, nb, size, acell, bcell );
      idcount := (if asize > bsize then asize else bsize) - size;
      if kbon ∨ kbonAGM then
      begin
         LINE;
         writetext( ⟊<AGM: i = ⟋ );
         writeinteger( ⟊-d⟋, i );
         writetext( ⟊<, twoPower = ⟋ );
         writeinteger( ⟊-d⟋, twoPower );
         writetext( ⟊<, w1size = ⟋ );
         writeinteger( ⟊-d⟋, w1size );
         writetext( ⟊<, prevw1size = ⟋ );
         writeinteger( ⟊-d⟋, prevw1size );
         writetext( ⟊<, A =⟋ );
         PRINT( A, asize, na );
         LINE;
         writetext( ⟊< compare size = ⟋ );
         writeinteger( ⟊-d⟋, size );
         writetext( ⟊<, idcount = ⟋ );
         writeinteger( ⟊-d⟋, idcount );
         writetext( ⟊<, B =⟋ );
         PRINT( B, bsize, nb )
      end;
      if prevw1size ≤ decimals then goto AGM LOOP END;
      prevw1size := w1size;
      SQUARE( W1, w1size, nw1, W2, w2size, nw2 );
      MULT( W2, w2size, nw2, twoPower );
      DIVIDE( W2, w2size, nw2, 4, empty );
      ADD( W2, w2size, nw2, 1, C, csize, nc );
      twoPower := twoPower + twoPower;
      COPY( A, asize, na, W1, w1size, nw1 );
      ADD( B, bsize, nb, 1, W1, w1size, nw1 );
      LONGMULT2( A, asize, na, B, bsize, nb, W2, w2size, nw2 );
      COPY( W1, w1size, nw1, A, asize, na );
      DIVIDE( A, asize, na, 2, empty );
      if idcount ≤ 0 then
      begin
         SQRT4( B, bsize, nb, W2, w2size, nw2, W1, w1size, nw1 )
      end
      else
      begin
         COPY( A, asize, na, B, bsize, nb );
         SQRT5( B, bsize, nb, W2, w2size, nw2, W1, w1size, nw1 )
      end
```

```
   end;
   AGM LOOP END:
   ftrack := ftrack - 2
end EXPISQN;
integer procedure PI TO 2( A, asize, na, B, bsize, nb, C, csize, nc );
comment A := pi using AGM;
value na, nb, nc;
integer asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   SQRT3( 0.5, B, bsize, nb, A, asize, na, C, csize, nc );
   ASSIGN( 1.0, A, asize, na );
   AGM( A, asize, na, B, bsize, nb, C, csize, nc );
   MULT( C, csize, nc, 4 );
   ASSIGN( 1.0, B, bsize, nb );
   ADD( C, csize, nc, -1, B, bsize, nb );
   MULT( A, asize, na, 2 );
   SQUARE( A, asize, na, C, csize, nc );
   LONGDIVIDE( C, csize, nc, B, bsize, nb, A, asize, na )
end PI TO 2;
integer procedure EXPISQN(N, A, asize, na, B, bsize, nb, C, csize, nc);
value N, na, nb, nc;
integer N, asize, na, bsize, nb, csize, nc;
integer array A, B, C;
begin
   integer xsize, nx;
   integer array X[0:c39];
   nx := ftrack;
   ftrack := ftrack + 1;
   PI TO(A, asize, na, B, bsize, nb, C, csize, nc);
   SQRT(N, B, bsize, nb, C, csize, nc, X, xsize, nx);
   LONGMULT(A, asize, na, B, bsize, nb, C, csize, nc);
   EXP(C, csize, nc, A, asize, na, B, bsize, nb);
   ftrack := ftrack - 1
end EXPISQN;
integer procedure FACTAB(from, step, to, A, asize, na);
value from, step, to, na;
integer from, step, to, asize, na;
integer array A;
begin
   integer N, n;
   ACCEPT(limit > 1 + 0.05×to×ln(to));
   ASSIGN(1, A, asize, na);
   for N := 2 step 1 until from -1 do
   MULT(A, asize, na, N);
   n := step -1;
   for N := from step 1 until to do
   begin
      MULT(A, asize, na, N);
      n := n + 1;
      if n = step then
      begin
         n := 0;
         LINE;
         writetext(⫷<N: ⫸);
         write integer(⫷-ddddd⫸, N);
         writetext(⫷<, FAC(N): ⫸);
         PRINT(A, asize, na)
      end if n
   end for N
end FACTAB;
integer procedure POWTAB1(from, step, to, a, A, asize, na);
value from, step, to, a, na;
integer from, step, to, a, asize, na;
```

```
integer array A;
begin
   integer N, n;
   ACCEPT(limit > 1 + 0.05×to×ln(a));
   LINE;
   writetext(⦃<a: ⦄);
   write integer(⦃-ddddddddddd⦄, a);
   ASSIGN(1, A, asize, na);
   for N := 1 step 1 until from -1 do
   MULT(A, asize, na, a);
   n := step -1;
   for N := from step 1 until to do
   begin
      MULT(A, asize, na, a);
      n := n + 1;
      if n = step then
      begin
         n := 0;
         LINE;
         writetext(⦃<N : ⦄);
         write integer(⦃-ddddd⦄, N);
         writetext(⦃<, a↑N: ⦄);
         PRINT(A, asize, na)
      end if n
   end for N
end POWTAB1;
integer procedure POWTAB2(from, step, to, b, A, asize, na);
value from, step, to, b, na;
integer from, step, to, b, asize, na;
integer array A;
begin
   integer N, n;
   ACCEPT(limit > 1 + 0.05×b×ln(to));
   LINE;
   writetext(⦃<b: ⦄);
   write integer(⦃-ddddddddddd⦄, b);
   for N := from step step until to do
   begin
      ASSIGN(1, A, asize, na);
      for n := 1 step 1 until b do
      MULT(A, asize, na, N);
      LINE;
      writetext(⦃<N: ⦄);
      write integer(⦃-ddddddddddd⦄, N);
      writetext(⦃<, N↑b: ⦄);
      PRINT(A, asize, na)
   end for N
end POWTAB2;
integer procedure ISOM(N, PRI, psize, np, SEC, ssize,
ns, TER, tsize, nt);
value N, np, ns, nt;
integer N, psize, np, ssize, ns, tsize, nt;
integer array PRI, SEC, TER;
begin
   integer sbase, nu, nv, usize, vsize, k, n, m, i, j, si, sj, q, sk, f;
   boolean empty;
   integer array U, V[0:c39];
   integer procedure size(n);
   value n;
   integer n;
   begin
      get(U, FREE, sbase + n:40);
      size := U[n mod 40]
   end size;
```

```
procedure store(n, size);
value n, size;
integer n, size;
begin
   get(U, FREE, sbase + n:40);
   U[n mod 40] := size;
   put(U, FREE, sbase + n:40)
end store;
large := true;
nu := ftrack;
nv := nu + 1;
f:=nv+1;
sbase:=1+step×(1+f+N);
ftrack:=f+1+N+1+N:40;
ASSIGN(1, PRI, psize, f);
store(0, psize);
for n := 1 step 1 until N do
begin
   ASSIGN(0, SEC, ssize, ns);
   ASSIGN(0, TER, tsize, nt);
   m := (n - 1):2;
   for i := 1 step 1 until m do
   begin
      j := n - 1 - i;
      si := size(i);
      sj := size(j);
      if i < j then
      LONGMULT(PRI, si,
      f + i, U, sj, f + j, V, vsize, nv)
      else
      begin
         ASSIGN(1, U, usize, nu);
         ADD(V, sj, f+ j, 1,
         U, usize, nu);
         LONGMULT(PRI, si,
         f + i, U, usize, nu, V, vsize, nv);
         DIVIDE(V, vsize, nv, 2, empty)
      end i ≥ j;
      ADD(V, vsize, nv, 1, SEC, ssize, ns)
   end for i;
   m := (n - 2):2;
   for i := 1 step 1 until m do
   begin
      j := n - 1 - 2×i;
      si := size(i);
      sj := size(j);
      ASSIGN(if i ≠ j then 0 else 2, U, usize, nu);
      ADD(PRI, sj, f + j, 1,
      U, usize, nu);
      LONGMULT(PRI, si,
      f + i, U, usize, nu, V, vsize, nv);
      ADD(U, usize, nu, 1, V, vsize, nv);
      LONGMULT(V, vsize, nv,
      PRI, si, f + i, U, usize, nu);
      DIVIDE(U, usize, nu, if i ≠ j then
      2 else 6, empty);
      ADD(U, usize, nu, 1, TER, tsize, nt)
   end for i;
   m := (n - 4):3;
   for i := 1 step 1 until m do
   begin
      q := (n - 2 - i):2;
      for j := i + 1 step 1 until q do
      begin
```

```
            k := n - 1 - i - j;
            si := size(i);
            sj := size(j);
            sk := size(k);
            LONGMULT(PRI,
            sj, f + j, U, sk, f + k, V, vsize, nv);
            LONGMULT(V, vsize,
            nv, PRI, si, f + i, U, usize, nu);
            ADD(U, usize, nu, 1, TER, tsize, nt)
         end for j
      end for i;
      LINE;
      writetext(⟨<N: ⟩);
      write integer(⟨-ddddddd⟩, n);
      LINE;
      writetext(⟨<PRI(N):⟩);
      PRINT(PRI, size(n - 1), f + n - 1);
      LINE;
      writetext(⟨<SEC(N):⟩);
      PRINT(SEC, ssize, ns);
      LINE;
      writetext(⟨<TER(N):⟩);
      PRINT(TER, tsize, nt);
      LINE;
      ADD(TER, tsize, nt, 1, SEC, ssize, ns);
      ADD(PRI, size(n - 1), f + n - 1, 1,
      SEC, ssize, ns);
      COPY(SEC, ssize, ns, U, usize, f + n);
      store(n, usize)
   end for n;
   ftrack := ftrack - (4 + N + N:40)
end IOSM;
real procedure clock count;
code clock count;
1, 37;
  zl , grf p-1 ; RF:=clock count; clock count:=0; stack[p-1]:=RF;
e;
procedure CALCULATE;
begin
   integer array A, B, C[0:c39];
   integer procedure next;
   begin
      integer x;
      if show then LINE;
      writetext(⟨<r := ⟩);
      x := read integer;
      if show ∨ showAll then write(⟨ddddddddd⟩, x);
      next := x
   end next;
   real procedure next real;
   begin
      real x;
      if show then LINE;
      writetext(⟨<r := ⟩);
      x := read real;
      if show ∨ showAll then write(⟨d.dddddd10-ddd⟩, x);
      next real:= x
   end next real;
   integer procedure STOP;
   go to EX;
   procedure ORDER(text, command);
   string text;
   integer command;
   begin
```

```
         integer dummy;
         type := type + 1;
         if type = TYPE then
         begin
            writetext(text);
            clock count;
            dummy := command;
            LINE;
            writetext( {<clock count: } );
            write({ddddddd.d}, clock count);
            go to NEW
         end if this type
      end ORDER;
      ftrack := 4;
NEW: LINE;
      LINE;
      writetext({<No: });
      TYPE := read integer;
      if show ∨ showAll then write ({dd}, TYPE);
      type := 0;
      ORDER({<< A := r;}, ASSIGN(next real, A, asize, 1));
      ORDER({<< write(A);}, PRINT(A, asize, 1));
      ORDER({<< B := A;}, COPY(A, asize, 1, B, bsize, 2));
      ORDER({<< C := A;}, COPY(A, asize, 1, C, csize, 3));
      ORDER({<< A := B;}, COPY(B, bsize, 2, A, asize, 1));
      ORDER({<< C := B;}, COPY(B, bsize, 2, C, csize, 3));
      ORDER({<< A := C;}, COPY(C, csize, 3, A, asize, 1));
      ORDER({<< B := C;}, COPY(C, csize, 3, B, bsize, 2));
      ORDER({<< A := A + B;}, ADD(B, bsize, 2, 1, A, asize, 1));
      ORDER({<< A := A - B;}, ADD(B, bsize, 2, -1, A, asize, 1));
      ORDER({<< A := A×r;}, MULT(A, asize, 1, next));
      ORDER({<< A := A/r;}, DIVIDE(A, asize, 1, next, empty));
      ORDER({<< C := A×B;}, LONGMULT(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< A := PI;}, PI TO(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< A := exp(B);}, EXP(B, bsize, 2, A, asize, 1, C, csize, 3));
      ORDER({<< A := sqrt(r);}, SQRT(next real, A, asize, 1, B, bsize, 2, C, csize, 3)
      ORDER({<< A := exp(PI×sqrt(r));},
      EXPISQN(next, A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< FACTORIAL TABLE(r, r, r);},
      FACTAB(next, next, next, A, asize, 1));
      ORDER({<< POWER TABLE(r, r, r, r↑variable);},
      POWTAB1(next, next, next, next, A, asize, 1));
      ORDER({<< POWER TABLE(r, r, r, variable↑r);},
      POWTAB2(next, next, next, next, A, asize, 1));
      ORDER({<< ISOMER TABLE(r);},
      ISOM(next, A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< stop}, STOP);
      ORDER({<< PRINT STATISTICS}, STATISTICS PRINT);
      ORDER({<< C := A<×2>B;}, LONGMULT2(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< (C,A) := (A:B,A mod B);}, LONGDIVIDE(A, asize, 1, B, bsize, 2, C, csiz
      ORDER({<< A := sqrt2(r);}, SQRT2(next real, A, asize, 1, B, bsize, 2, C, csize,
      ORDER({<< A := sqrt3(r);}, SQRT3(next real, A, asize, 1, B, bsize, 2, C, csize,
      ORDER({<< A := TO REAL(A);}, ASSIGN(TO REAL( A, asize, 1), A, asize, 1));
      ORDER({<< A := sqrt(B);}, SQRT4(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< (A,B,C) := AGM(A,B);}, AGM(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< A := PI 2;}, PI TO 2(A, asize, 1, B, bsize, 2, C, csize, 3));
      ORDER({<< A := sqrt5(B) starting at A;}, SQRT5(A, asize, 1, B, bsize, 2, C, csiz
      ORDER({<< (C,A) := (A:B,A mod B) with r decimals;}, LONGDIVIDE2(A, asize, 1, B,
      ORDER({<< A := sqrt6(B) starting at A;}, SQRT6(A, asize, 1, B, bsize, 2, C, csiz
      go to NEW;
EX:end CALCULATE;
   STATISTICS INIT;
   linerest := 69;
   MODUL := 10000000000;
```

```
   select(17);
   LINE;
   writetext( ┤<2011-Aug-03 18.09 / TN┤ );
   SELECT LANGUAGE;
   LINE;
   WRITE TEXT(
   ┤<PROGRAM DEMON-5. Beregning af store tal. Programmet simulerer en maskine med
3 registre, A, B og C, som har D decimaler og E cifre før kommaet.
Der anvendes følgende ordresystem:┤,
   ┤<PROGRAM DEMON-5. Calculation of large number. The program simulates a computer
with 3 registers, A, B, and C, with D decimals and E integer digits.
The following command system is used:┤,
   ┤<PROGRAMME DEMON-5. Calcul des nombres eleves. Le programme simule une machine a
3 registres, A, B, et C, avec D decimales et E chiffres entiers.
On utilise les commandes suivantes:┤,
   ┤<PROGRAMM DEMON-5. Berechnung von grossen Zahlen. Das Programm simuliert eine
Maschine mit 3 Registern, A, B, und C, mit D dezimalstellen und E Ganzzahlstellen.
Man verwendet die folgende Befehle:┤);
   linerest := linerest - 2;
   LINE;
   LINE;
   writetext(┤<
  No:
   1: A := typein; 13: C := A×B;
   2: write(A); 14: A := PI;
   3: B := A; 15: A := exp(B);
   4: C := A; 16: A := sqrt(typein);
   5: A := B; 17: A := exp(PI×sqrt(r));
   6: C := B; 18: A := table of factorial function;
   7: A := C; 19: A := table of a↑N;
   8: B := C; 20: A := table of N↑b;
   9: A := A + B; 21: A := table of alcohol isomers;
  10: A := A - B; 22: STOP;
  11: A := A×typein; 23: print statistics
  12: A := A/typein; 24: C := A<×2>B;
                         25: (C,A) := (A:B,A mod B)
                         26: A := sqrt2(typein);
                         27: A := sqrt3(typein);
                         28: A := TO REAL(A);
                         29: A := sqrt(B);
                         30: (A,B,C) := AGM(A,B);
                         31: A := PI 2;
                         32: A := sqrt5(B); starting at A
                         33: (C,A) := (A:B,A mod B) with typein decimals
                         34: A := sqrt6(B); starting at A
┤);
   linerest := linerest - 14;
   LINE;
   WRITE TEXT(
   ┤<Vi lader først maskinen demonstrere nogle eksempler:┤,
   ┤<We first let the computer show some examples:┤,
   ┤<La machine nous donne d abord quelques examples:┤,
   ┤<Zuerst zeigt die Maschine einige Beispiele:┤);
   show := true;
   showAll := true;
   comment MK: where moved;
   where(┤<free┤, FREE);
   if true then
   begin
   select(16);
   for D := read integer while D ≥ 0 do
   begin
      LINE;
      writetext(┤<D:┤);
```

```
      writeinteger(╞-ddddddd╡, D);
      decimals := D;
      if decimals > 0 then
      decimals := -((decimals-1):10+1);
      E := read integer;
      LINE;
      writetext(╞<E:╡);
      writeinteger(╞-ddddddd╡, E);
      limit := (E-1):10;
      step := (limit - decimals):40 + 1;
      large := step > 1;
      c39 := if large then 39 else limit - decimals;
      CALCULATE;
E1:end for decimals;
   LINE
   end;
   LINE;
   show := false;
   select(17);
   WRITE TEXT(
   ╞<Nu kan De forsøge:╡,
   ╞<Now you may try:╡,
   ╞<Maintenant vous pouvez essaier:╡,
   ╞<Jetzt koennen Sie versuchen:╡);
   for D := ASK NUMBER(
   ╞<Opgiv antal decimaler, D. -1 er stop╡,
   ╞<Specify number of decimals, D. -1 is stop╡,
   ╞<Specifiez le nombre de decimales, D. -1 est termination╡,
   ╞<Bitte, die Anzahl von Dezimalstellen, D, angeben. -1 is Schluss╡)
   while D ≥ 0 do
   begin
      if showAll then
      begin
         writeinteger( ╞-d╡, D )
      end;
      decimals := D;
      if decimals > 0 then
      decimals := -((decimals-1):10 + 1);
      E := ASK NUMBER(
      ╞<Og antallet af heltalscifre, E╡,
      ╞<And the number of integer digits, E╡,
      ╞<Et le nombre de chiffres entiers, E╡,
      ╞<Und die Anzahl der Ganzzahlstellen, E╡);
      if showAll then
      begin
         writeinteger( ╞-d╡, E )
      end;
      limit := (E-1):10;
      step := (limit - decimals):40 + 1;
      large := step > 1;
      c39 := if large then 39 else limit - decimals;
      CALCULATE;
E2:end for decimals
end
t<
Ff
380, 20
16, 2, 2
26, 2, 2
27, 2, 2
1, 2
29, 2, 2
22, -1,
```