(1)

```
let Parse(C, Salt) = C 1 (C, Salt)

in
let Seq(C, Salt) = Parse(C 3, Parse(C 2, Salt))

and Or(C, Salt)  = valof (   res Parse(C 2, (Salt 1, L));
                           L: res Parse(C 3, Salt)   )

and Qu(C, Salt)  = let S, Alt = Salt 1, Salt 2
                   in C 2 eq Stem S -> (Stern S, Alt) | (goto Alt)

in

let A, B, C = 0, 0, 0
in

A := (Or, (Seq, B, C), (Qu, '2'));

B := (Or, (Qu, '0'), (Seq, (Qu, '0'), C));

C := (Seq, (Qu, '1'), (Seq, A, (Qu, '1')));


Loop:

( let Ch, S = 0, '' in

Write 'Please type some input*n*n' ;

L: Ch := Readch nil;

   if Ch eq '*n' do ( if S eq '' do Finish nil;
                      Parse(A, (S, Fail));
                      Write(S, '  is OK*n' );
                      goto Loop;

                Fail: Write(S, '  is not OK*n*n' );
                      goto Loop    );

   S := S %Conc Ch;
   goto L    )


// end of program
```

R 3.733+.850

(2)

```
2
2  is OK
Please type some input

00101211
00101211  is not OK

Please type some input

0121
0121  is OK
Please type some input

0101211
0101211  is OK
Please type some input
```

```
Execae$oh fanrs##edsy@   qed aesth a syn pal
@@@3$0o&ups          3507 cycles
```

```
def Innerproduct(a, b) = valof
       $( let i = Order a
          and rec f n = n ls 1 -> 0 !
                        a n * b n + f(n-1)
          in
          if i eq Order b then f i ;
          do Write 'Illegal call to *'Innerproduct*'';
          res SYSTEMERROR 0  $)

and A1 = 1,2,3
and A2 = 4,5,6
and B1 = 6,7
and B2 = 9,10
in

def Test(p, q) be
       do Write(p,'*n', q, '*n*t', Innerproduct(p, q), '*n')
in

do Test(nil, nil);
do Test(A1, A2);
do Test(B1, B2);

do Write 'and now an illegal argument  ...*n*n' ;
do Test(A1, B1);

do Write 'All done.*n'




// Next example

def Em S = S eq ''
in

def Substring(a, b) =
          f(a, b, true)
          where rec f(x, y, B) =
                    Em x -> true!
                    Em y -> false!
                    Stem x eq Stem y -> f(Stern x, Stern y, false)!
                    B -> f(a, Stern b, true)!
                    false
in


def Test(a, b) be
          def Qu S = Q %Conc S %Conc Q
          and N, Q = '*n', '*''
          in
          do Write(Qu a, N, Qu b, N, '*t', Substr(a, b), N, N)
in
do Test('', 'ab');
do Test('ab', '');
do Test('abcd', 'abcd');
do Test('bc', 'abcd');
do Write '*nAll done.*n'
```

```
// Next example

def Cycle A = valof
        $( def n, i = Order A, 1
           and T = nil
           in
           if n eq 0 res false;

        L: T := T aug ( A i ls 1 -> n+1!
                        A i gr n -> n+1!
                        $ A i    );
           i := i + 1;
           if i ls n then goto L;

        M: i := i + 1;
        N: i, T i := T i, n;
           if i ls n then goto N;
           if i eq n then res true;
           n := n - 1;
           if n gr 0 then goto M;
           res false                          $)
in
do Test Cycle;
do Write '*nAll done.*n'

R 3.966+2.083
```

```
READY.

i
W 1657.7
 LOGIN PLEASE.
READY.

login t338 chards
W 1658.7
Password
 T0338  5162 LOGGED IN  12/22/67    1700.f OUIT,
P .016+.000

r eda mrdbg pal
W 1700.4
Old file (INPUT FILE) --Do you wish to delete it?  yes
Edit
1 res
                                res nil
1 (f
                Pterm(f) = Symb='V' logor Symb='C' logor Symb='(' -*
1 (f
                                                Pterm(f Rbasic()),
c /Rbasic()/(Rbasic())/
                                                Pterm(f (Rbasic())),
file
*
P 4.100+2.366

*r pal mrdbg pocode
W 1703.0
Pal compiler entered
Pal loader entered
Execution
Debug called
*Write(1,2,3,4,5);
12345$$$
*Share(x,x);
true
*Share(1,1);Share(x,y);Share(f,f)##e);
false
false

Run time error: Undeclared name f

Run time error: ERROR function destroyed

Run time error: Undeclared name e

Run time error: ERROR function destroyed
false
*{.
*{x;
nil
Program re-entered
Debug called
2
* x.x.x.x.
2
Program re-entered
Debug called
nil
Program re-entered
Debug called
nil
```

(2)

```
Program re-entered
Debug called
nil
Program re-entered
Debug called
*x. x. x.
6
Program re-entered
Debug called
7
Program re-entered
Debug called
8
Program re-entered
Debug called
*D#Hd(Cons(Cons(x, Cons(y,2)),z));

Run time error: Undeclared name Cons

Run time error: ERROR function destroyed

Run time error: Undeclared name Cons

Run time error: ERROR function destroyed

Run time error: Undeclared name Cons

Run time error: ERROR function destroyed

Run time error:  nil  applied to ( 2  2 )

Run time error:  nil  applied to ( 9  nil )

Run time error:  nil  applied to ( nil  3 )

Run time error: Argument structure  nil  incorrect

Run time error: ERROR function destroyed.

Run time error: Undeclared name x

Run time error: ERROR function destroyed
nil


*x;y;z;
9
2
3
*(123,456,789) 2, (123, (456, 789), 'asd') 2 3;

Run time error: ( 456  789 ) applied to  3
(456, (456, 789))
 QUIT,
R 31.216+12.116
```

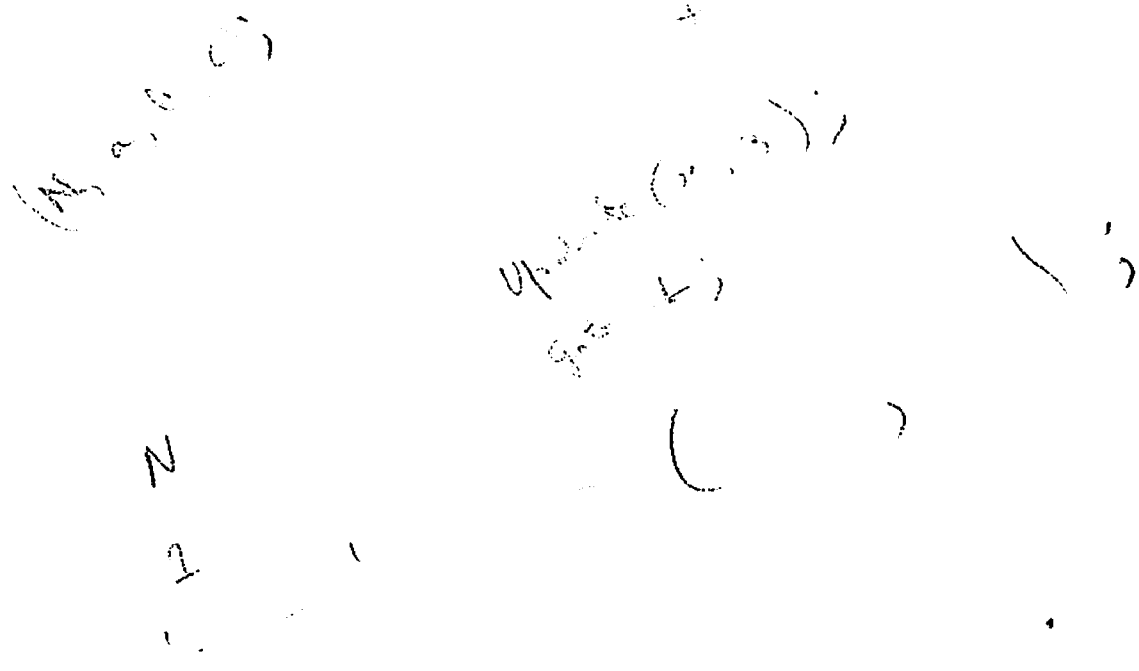
VI N

```
M DF10t0 mrdbr pal
```

(3)

```
def Write x = Istuple x -* W(1, Length x), Print x
              where rec W(i, n) = n=0 -* Print nil,
                                  i gr n -* dummy,
                                  ( Print(x i); W(i+1, n)  )

def Debug() =
        ( let j = ji in
          let Lookup S = Lookupind(S, j)
          in
          Write('Debug called*n');
          let Ch, Symb, Val =0, 0, 0
          in
          let Chkind(x) =
                    x='0' -* 0, x='1' -* 1, x='2' -* 2, x='3' -* 3,
                    x='4' -* 4, x='5' -* 5, x='6' -* 6, x='7' -* 7,
                    x='8' -* 8, x='9' -* 9,
                    x='*n' -* 10, x='*s' -* 10, x='*t' -* 10,
                    x=';' -* 11, x=',' -* 11, x='(' -* 11,
                    x=')' -* 11, x='.' -* 11,
                    x='*'' -* 12,
                    -1
          in
          let Nextsymb() =
                  ( let N, Kind = 0, 0 in

                L: Kind := Chkind(Ch);
                   Kind=10 -* ( Ch := Readch(); goto L),
                   Kind=11 -* ( Symb := Ch;
                                Ch := Readch() ),
                   Ch='*'' -*
                           ( Symb, Val := 'C', '' ;

                       Nsch: Ch := Readch();
                             Ch='*'' -* (Ch := Readch();
                                         goto Return  ),
                             Ch='**' -* ( Ch := Readch();
                                          Ch := Ch='t' -* '*t',
                                                Ch='n' -* '*n',
                                                Ch='s' -* '*s',
                                                Ch='b' -* '*b',
                                                Ch ), dummy;
                                Val := Val %Conc Ch;
                                goto Nsch ),
                    ( Symb, Val := 'C', '';
                   M: Kind Is 10 -* ( Val := Val %Conc Ch;
                                      Kind Is 0 -* ( Symb := 'V' ),
                                                    ( N := N*10 + Kind );
                                      Ch := Readch();
                                      Kind := Chkind(Ch);
                                      goto M  ),
                      Symb='C' -* ( Val := N ),
                Return: dummy  )  )
          in
          let rec (
              Pbasic() =
               val( let A = Symb='V' -* Lookup(Val),
                            Symb='C' -* $ Val,
                            Symb='(' -* ( Nextsymb();
                                          Rexp()  ),
                            res nil
                    in
                    Nextsymb();
                    --- A  )
```

```
            and
            Pterm(f) = Symb='V' logor Symb='C' logor Symb='(' -*
                                    Pterm(f (Phasic())),
                                    f

          and
            Pexp() =
               (  let A = nil in
               L: A := A aug Pterm(Phasic());
                  Symb=',' -* ( Nextsymb(); goto L ),
                  Length A = 1 -* A 1, A  )   )
        in
        Ch := Readch();
      L: Nextsymb();
        Write(Pexp(), '*n');
        Symb='.' -* Write('Program re-entered*n'), goto L  )

def ( x,y,z = 1, 2, 3)

L: Debug(); x := x + 1; goto L

p 3.966+1.833
```



(1)

```
//  This program solves the Gap Test for random-ness of a sequence
//  of decimal digits.  The function Random() is called successively
//  to return an integer between 0 and 9, inclusive.  The mean and
//  variance of the gap between successive occurrences of each digit
//  is calculated.

//  The number of observations to take is read in from the console.

//  This program was last modified on 01/07/68 at 16:28 by Evans.

let
      p, x = 99989, 54321
   within
   Random() =
      x := x + x;
      (x > p)  ->  (x := x - p)!  dummy;
      x - 10*(x/10)
in



let Readint()  =
//  Executing Readint() will return an integer typed on the console.
//  Readint accepts a line, and returns that value formed by
//  considering only the digits on the line.
val (
     let Num, Next, t, Sign = 0, nil, nil, 1
     in

     Loop:  // Come back here for each character.
     Next := Readch();  // The next character from the line.
     Next = '*n'  ->  (res Num*Sign)!  // Quit if newline read
     Next = '-'   ->  (Sign := -1)!  // Read a negative number
     t := StoI Next;  // Convert character to integer form.
     (t > -1) & (t < 10)  -> (Num := 10*Num+t)! dummy;
     goto Loop
    )
in

let Write x =  // write a tuple without commas or outer parens
   Istuple x
      -> W(1, Length x)
      !  Print x
   where
   rec W(i, n) =
      (n = 0)
         -> Print nil!
      (i > n)
         -> dummy!
      (
```

```
            Print(x i);
            W(i+1, n)
        )
in


let Data = nil  // this will be updated to a 10-tuple of 4-tuples
in

let Test n =  // Test calls Random 'n' times, and updates Data.

   let i, j, t = 1, nil, nil  // set i to 1, and create j and t
   in

   Loop:  // this loop calls Random 'n' times
      let Cell = Data(Random() + 1)  // Cell is the relevant 4-tuple
      in
      Null Cell  // test if this is first occurrence of this digit
      -> // it is, so initialize
         (
            Cell := $i, 1, 0, 0
         )
      !
         (
            j := i - Cell 1;  // length of this gap
            Cell 1 := i;  // index of last occurrence
            Cell 2 := Cell 2  +  1;  // count occurrences
            Cell 3 := Cell 3  +  j;  // sum of gaps
            Cell 4 := Cell 4  +  j*j // sum of squares of gaps
         )
      ;
      i := i + 1;  // count observations
      (i > n)  ->  dummy!  goto Loop
in

let Printresults() =  // the function that prints results
   let i, Count, Mean, Variance = nil, nil, nil, nil
   in
   // Calculate means and variances, and print results.

   Write ('n count*t   mean          variance*n*n');
   i := 1;  // count from 1 to 10

   PrLoop:  // the loop


   let Cell = Data i  // the relevant cell
   in

   Null Cell
   ->
      Write (i-1, '   0*t   no observations of this digit*n')
   !
   (
      Count := ItoR(Cell 2);  // count number of observations
      Mean := ItoR(Cell 3) / Count;
```

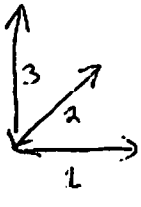
```
      Variance := ItoR(Cell 4) / Count  -  Mean*Mean;
      Write(i-1, '    ', Cell 2, '*t', Mean, Variance, '*n')
   );
   i := i + 1;
   (i < 11) -> goto PrLoop! Write ('*n*n')
in



// Here (finally) is the program...
Start:
Write ('*nType a number*n');
let n = Readint()  // read an integer from the console
in

(n > 0)
->
   (
      Write ('*n', n, ' observations will be taken.*n*n');
      Data := nil, nil, nil, nil, nil, nil, nil, nil, nil, nil;
      Test n;
      Printresults();
      goto Start
   )
!
(n < 0)
->
   (Debug(); goto Start)
!
   Write ('*nAll done.*n')
```

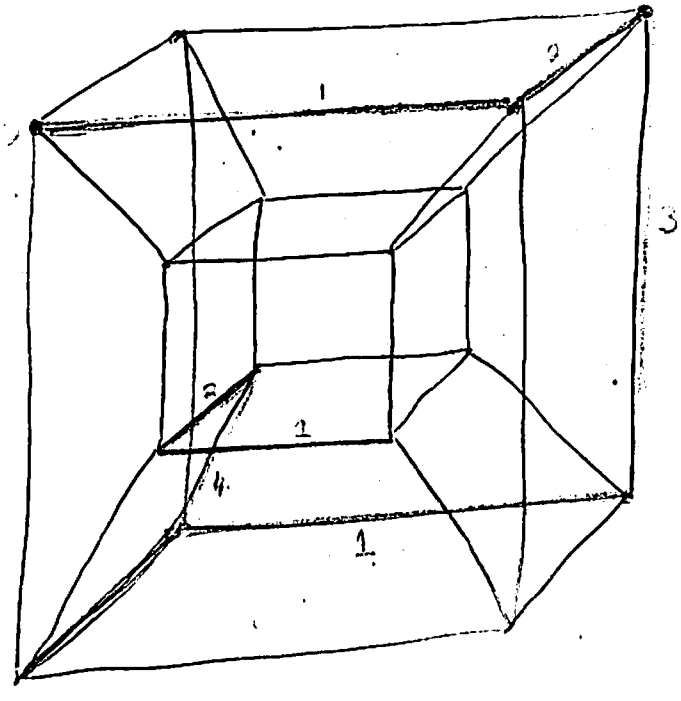
```
// PAL demonstration -- tuple-producing functions.

// Last modified on 11/29/67 at 16:30 by Evans.

// Copied from a program of Martin Richards

let I x  =  x  //  the 'identity' function
in

let Tuple n  =  // The definition of Tuple is such that  Tuple 3 x y z
                // makes a 3-tuple whose elements are x, y and z.
     T n I
     where rec T n =
          n=0  -*  I,
          ll h a.  T (n-1) (ll s. h s a)
in

let Node3  =  Tuple 3
in

let  H1 x y z  =  x
and  H2 x y z  =  y
and  H3 x y z  =  z
in

let  A, B  =  Node3 1 2 3, Node3 4 5 6
in

let Outit(A, B)  =  (Print(A H1, A H2, A H3, B H1, B H2, B H3);  Print('*n') )
in

Outit(A, B);

A H1, B H2  :=  6, 9;

Outit(A, B);

let a, b, c  =  1, 2, 3
in

let A, B  =  Node3 a b c, Node3 c b a
in

Outit(A, B);

A H1  :=  5;

Outit(A, B)
```

```
r printa 3tuple pal
W 1826.2

3TUPLE    PAL      11/27/67  1826.3

// PAL demonstration -- tuple-producing functions.
// Special in this case to 3-tuples.

// Last modified on 11/27/67 at 17:53 by Evans.

// Copied from a program of Martin Richards

let  Node3 a b c  =  11 S. S a b c
in

let  3H1 x y z  =  x
and  3H2 x y z  =  y
and  3H3 x y z  =  z
in

let  A, B  =  Node3 1 2 3, Node3 4 5 6
in

let Outit()  =  (Print(A 3H1, A 3H2, A 3H3, B 3H1, B 3H2, B 3H3);  Print('*n')
in

Outit nil;

A 3H1, B 3H2  :=  6, 20;

Outit nil

R 1.133+1.200

r cpal 3tuple
W 1827.2
Pal compiler entered
Pal loader entered
Execution
(1, 2, 3, 4, 5, 6)
(6, 2, 3, 4, 20, 6)

Execution finished
R 2.750+4.716
```

A = λ S. S 1 2 3

```
r printa ncubes pal
W 1338.3

NCUBES    PAL     12/03/67  1338.4

let N = 4 in

let rec g n y m =  m=n -* not y m, y m
in

let rec Next (n, y, x, B) = n=0 -* not B,
                            y n = x n -* Next(n-1, y, x, B),
                            B -* Next(n-1, y, x, false),
                            false
in

let rec Adj(y, n, P, x) =
                      Next(N, y, x, true) -* true,
                      n=0 -* false,
                      Adj(g (P n) y, n-1, P, x)
in
let Ok(n, P, k, x) = not Adj(g (P n) x, n-1, P, g k x)


let  rec Srch(n, k, P, x) =
        k gr N -* 0,
        ( Ok(n, P,  k, x) -*(Srch(n+1, 1, Q, g k x) where Q =
                                                  let Q = Aug P in
                                                  Q (n+1) := k;
                                                  Print Q; Print '*n';
                                                  Q),

                                dummy;
          Srch(n, k+1, P, x)     )
in
Srch(3, 1, (1,2,3), (false, false, false, true))

R 1.750+1.333

r pal ncubes
W 1340.6
Pal compiler entered
Pal loader entered
Execution
(1, 2, 3, 1)
(1, 2, 3, 1, 4)
(1, 2, 3, 1, 4, 2)
(1, 2, 3, 1, 4, 2, 1)
(1, 2, 3, 1, 4, 3)
(1, 2, 3, 4)
(1, 2, 3, 4, 1)
(1, 2, 3, 4, 1, 2)
(1, 2, 3, 4, 1, 3)
(1, 2, 3, 4, 2)
(1, 2, 3, 4, 2, 1)

Execution finished
R 49.833+8.800
```





```
M 1557.2

PALIPE    PAL      12/02/67  1557.2

def Hd(x, y) = x

def Tl(x, y) = y

def ( Applypath P S =
    let K = Length P in
    T(S, 1) where rec T(s, n) = n gr K -* s,
                                T(s(P n), n+1) )

def Order x = Length x

def Cy S = let List = nil in Copy S

          where Lookup Nde =
               let rec Lkp L = Null L -* ( let NewN = nil in
                                           List := ((Nde, NewN), $ List);
                                           false, NewN    ) ,
                                Share(Nde, Hd L 1) -* (true, Hd L 2) ,
                                Lkp(Tl L)
               in Lkp List

   within rec Copy Node = let End, CpyN = Lookup Node in
          End -* CpyN ,
          not Istuple Node -* ( CpyN := Node; CpyN ) ,
          let i, Size = 1, Order Node in
   CpyIp: i gr Size -* CpyN ,
          ( CpyN := Aug CpyN;
            CpyN := Swing(CpyN, i, Copy(Node i));
            i := i+1;
            goto CpyIp     )

P 1.533+1.316
```

```
JTEST     PAL      12/12/67  1710.0

Print (   let a = 0 in

          let b = 2
          within J = jj
             and f x = x, 11 S. S+b
          in
          a=0 -* (a := 1; J f 3), f J )

R .483+1.933

r pal jtest
W 1710.4
Pal compiler entered
Pal loader entered
Execution
5
Execution finished
R 2.783+5.133
```

TP5 PAL 09/13/87 1530.3

```
let Tuple n = T n (11 c.c)
    where rec T n = n=0 => (11 c. 11 c. c c),
                     11 b. 11 a. T(n-1)( 11 c. b c c)

in
let Node3 = Tuple 3
in
let H1 x y z = x
and H2 x y z = y
and H3 x y z = z

in
let A, B = Node3 1 2 3, Node3 4 5 6

in Print(A H1, A H2, A H3, B H1, B H2, B H3); Print("\n");

   A H1, B H2 := 5, 20;

   Print(A H1, A H2, A H3, B H1, B H2, B H3)
```

```
1 2 3 4 5 6
1 20 3 4 5 6
```

// H1 = λx. λy. λz. x

// A = λS.((S (1))(2))(3)

λ

H3 is a selector function.

B is a node

Co-Routine-Maker devised by Art Evans Jr.

```
let CRM(Syn) =

    let FL, GL, t = 0, 0, 0 in

    let y x = ( GL, t := L, x;
                goto FL;
              L: dummy )

    and f() = ( FL := L;
                goto GL;
              L: t )

    in
    GL := L;
    Syn(f);
    L: g

in
let Lex(Sym) = (
        --- Read().
        Syn(Symbol)
        Syn( ... )
        ... )

and Syn(Lex) = ( ... Lex() ... Lex() ... )

in Lex( CRM(Syn) )
```

// Lex is a
// a lexical analyser passing
// symbols to Syn one at a
// time

// Syn is a syntax
// analyser obtaining symbols
// by calls for the function Lex