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 -> (Stem 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*nn';
L: Ch := ReaDCh nil;
   if Ch eq '*n' do ( if S eq '' do Finish nil;
                    Parse(A, (S, Fail));
                    Write(S, ' is OK*nn');
                    goto Loop;
                    Fail: Write(S, ' is not OK*nn');
                    goto Loop       )

   S := S %Conc Ch;
   goto L    )

// end of program
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
def innerproduct(a, b) = valof
$ if i = Order a
    and rec f n = n ls 1 \rightarrow 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 \rightarrow true!
        Em y \rightarrow false!
        Stem x eq Stem y \rightarrow f(Stern x, Stern y, false)!
        B \rightarrow 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$ )$

    do Test Cycle;
    do Write '*nAll done.*n'

R 3.966+2.083
1 1657.7
LOGIN PLEASE.
2
login t338 chards
2 1658.7
Password
T0338 5162 LOGGED IN 12/22/67 1700.6 QUIT,
2 .016+.000
r redo m338 pal
2 1700.4
old file (INPUT FILE) -- Do you wish to delete it? yes
edit
1 res
res nil
1 (f)
Pterm(f) = Symb='V' logical Symb='C' logical Symb='!' -*
1 (f)
@ /P3asic()/(/P3asic())/ Pterm(f P3asic()),
file
*
P 4,100+2.366

r pal m338 pocode
2 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: ERROP function destroyed
Run time error: Undeclared name e
Run time error: ERROP function destroyed
false

|x|
nil
Program re-entered
Debug called
2
x.x.x.x.x.
2
Program re-entered
Debug called
nil
Program re-entered
Debug called
nil

2 1700+2.366

f
Program re-entered
Debug called

Program re-entered
Debug called

Program re-entered
Debug called

Program re-entered
Debug called

NIL(*cons(*cons(x, *cons(y, 2)), z));

Run time error: Undeclared name Cons
Run time error: EPPOR function destroyed
Run time error: Undeclared name Cons
Run time error: EPPOR function destroyed
Run time error: Undeclared name Cons
Run time error: EPPOR 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: EPPOR function destroyed.
Run time error: Undeclared name x
Run time error: EPPOR 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,
0 31.216+12.116

*AP1BTo medium 01
def Write x = I staple x ** W(1, Length x), Print x
    where rec W(i, n) = n=0 ** Print nil,
          i or 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 on');
      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='*' ** 10, x='*s' ** 10, x='*t' ** 10,
        x='*' ** 11, x='1' ** 11, x='(' ** 11,
        x='*' ** 11, x='1' ** 11,
        x='*' ** 12,
        -1
      in
      let Nextsym() =
        ( let N, Kind = 0, 0 in
          L: Kind := Chkind(Ch);
          Kind=0 ** ( Ch := Readch(); goto L ),
          Kind=1 ** ( Symb := Ch;
                        Ch := Readch() ),
          Ch='*1' **
              ( Symb, Val := 'a', '' ;
                Nsch: Ch := Readch();
                Ch='*1' ** ( Ch := Readch();
                                goto Return ),
                Ch='*1' ** ( Ch := Readch();
                            Ch := Ch='t' ** '1t',
                            Ch='n' ** '1n',
                            Ch='s' ** '1s',
                            Ch='h' ** '1h',
                            Ch ), dummy ),
          Val := Val % Conc Ch;
          goto Nsch ),
          ( Symb, Val := 'y', '' ;
          M: Kind 10 ** ( Val := Val % Conc Ch;
                        Kind 0 ** ( Symb := 'y' ),
                        ( N := N*10 + Kind ) ;
                        Ch := Readch();
                        Kind := Chkind(Ch);
                        goto M ),
          Symb='c' ** ( Val := N ),
          Return: dummy ) )
      in
      let rec ( Phasic() =
          val( let A = Symb='y' ** Lookup(Val),
                Symh='c' ** $ Val,
                Symh='(' ** ( Nextsym();
                              Rexp() ),
                res nil
          in
          Nextsym();
          -- -- ' 
        )
      )
    )
and
\[ pterm(f) = \text{Symh='V', logor Symh='C', logor Symh='(', f * pterm(f (bthesis()))),} \]

and
\[ pexp() = \]
\[
( \text{let A = nil in}
! : A := A aug pterm(bthesis());
Symh='!', -- (Nextsymh(); goto 1),
Length A = 1 -- A 1, A )
\]

In
Ch := Readch();
L: Nextsymh();
Write(pexp(), '*p');
Symh='.' -- Write('Program re-entered*pn'), goto L.

def ( x, y, z = 1, 2, 3)
L: Debup(); x := x + 1; goto L
n 3.966+1.833
This program solves the Gap Test for randomness 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
\[ p, x = 93989, 54321 \]
within

\begin{align*}
\text{Random()} &= \\
&= \begin{cases} 
  x := x + x; \\
  (x > p) \to (x := x - p)! \text{ dummy}; \\
  x = 10^*(x/10)
\end{cases}
\end{align*}

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

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

in

let Write x = // write a tuple without commas or outer parens

\begin{align*}
\text{Istuple x} \\
& \to W(1, \text{Length x}) \\
! \text{ Print x}
\end{align*}

\text{where}

\begin{align*}
\text{rec W(i, n) =} \\
(n = 0) & \to \text{ Print nil!} \\
(i > n) & \to \text{dummy!}
\end{align*}
Print(x i);
"(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 := Ito+(Cell 4) / Count - Mean*Mean;
Write(i-1, ', ', Cell 2, ', ', Cell 1, ', ', Mean, ', ', Variance, ', ')
}
i := i + 1;
(i < 11) -> goto PrLoop!

in

// Here (finally) is the program...
Start:
Write ('*nType a number*');
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')
let I x = x // the 'identity' function

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 1
            where rec T n =
                n=0 -* 1,
                11 h a. T (n-1) (11 s. h s a)
in
let Node3 = Tuple 3

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

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

let OutIt(A, R) = (Print(A H1, A H2, A H3, R H1, R H2, R H3); Print('*n'))
in

OutIt(A, R);

A H1, R H2 := 6, 9;

OutIt(A, R);

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

let A, P = Node3 a h c, Node3 c h a
in,

OutIt(A, P);

A H1 := 5;

OutIt(A, P)
let Node3 a b c = 1 1 1 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, R = Node3 1 2 3, Node3 4 5 6
in

let Output() = (Print(A 3H1, A 3H2, A 3H3, R 3H1, R 3H2, R 3H3); Print('n'))
in

Output nil;

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

Output nil

A 1.133, 1.200

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

Execution finished
P 2.750, 4.716
let rec Next (n, y, x, R) = n=0 -> not P,
y n = x n -> Next(n-1, y, x, R),
R -> Next(n-1, y, x, false),
false

let rec Adj(y, n, P, x) =
    Next(N, y, x, true) -> true,
n=0 -> false,
    Adj(x (P n) y, n-1, P, x)

let Ok(n, P, k, x) = not Adj(x (P n) x, n-1, P, k) x

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

Srch(3, 1, (1,2,3), (false, false, false, true))

P  1.750+1.333

r pal ncuhes
' 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
P  49.833+8.800
def "h(x, y) = x"
def TI(x, y) = y
def \( T(S, 1) \) where rec \( T(s, n) = n \text{ or } k = s \),
\( T(s(P n), n+1) \)
def Order x = Length x
def Cy S = let List = nil in Copy S
    where Lookup Nde =
        let rec Lkp L = Null L =\( T = \text{Coeff} \)
        list := ((Nde, NewM), $ list);
        Share(Nde, HD L 1) = (true, HD L 2),
        Lkp(T L L)
in Lkp List
    within rec Copy Node = let End, CpyN = Lookup Node in
        End =\( T = \text{Coeff} \)
    not IsUnique Node = ( CpyN := Node; CpyN )
        let i, Size = 1, Order Node in
    Cyln: i < Size =\( T = \text{Coeff} \)
        ( CpyN := S(int(CpyN, i, Copy(Node i))); i := i+1; goto Cyln )

C 1.533+1.316
Print ( let a = 0 in
    let b = 2
    within J = jj
    and f x = x, 11 S. 5+h
    in
    a = 0 -* (a := 1; J f 3), f J )

R .483+1.933
r pal itest
W 1710.4
Pal compiler entered
Pal loader entered
Execution ✔
Execution finished
R 2.783+5.133
H3 is a selector function.
B is a node.