```
comment fpi is the square root of (pi/2) and Least is set as a
  lower limit on x;

fpi := 1.253314137; Least := - 22.9;
if x = 0.0 then RMills := 1.0 / fpi else
if x < Least then RMills := 0.0 else
  begin integer d; real a, b, r, s, t, A0, A1, A2, B0, B1, B2;
  d := sign(x); x := abs(x);
  if x ≤ 2.0 then
    begin
    s := 0; a := 1.0;
    r := t := x; b := x X x;
    for a := a + 2.0 while s ≠ t do
      begin
      s := t; r := r X b / a;
      t := t + r
      end ;
    RMills := 1.0 / (fpi X exp(0.5 X x X x) - d X t)
    end
  else
    begin
    a := 2.0; r := s := B1 := x;
    A1 := x X x + 1.0; A2 := x X (A1 + 2.0);
    B2 := A1 + 1; t := A2 / B2;
    for a := a + 1.0 while r ≠ t ∧ s ≠ t do
      begin
      A0 := A1; A1 := A2;
      A2 := x X A1 + a X A0; B0 := B1;
      B1 := B2; B2 := x X B1 + a X B0;
      r := s; s := t;
      t := A2 / B2
      end ;
    RMills := if d = 1 then t else
      t / (2.0 X fpi X exp(0.5 X x X x) X t - 1.0)
    end
  end
end of RMills
```

Remark AS R1

A Remark on Algorithm AS 1 Sub-routine Package

By J. C. Gower

Rothamsted Experimental Station

In the subroutine package of AS 1 the procedure *checkset* (AS 1.1), as written, has some side-effects that can lead to inelegant features in programs using the package. For a table with n factors the procedure call *checkset* $(i, ifault)$ converts a factor number i to factor set form $2 \uparrow (n-i)$, and removes the factor set indicator *FS* whenever i is already in factor set form. The procedure also checks that no factor number larger than n is referred to. The side-effects are caused by calling i by name, and using it as both an input and output parameter. The procedures *scan* $(i, j, \ldots)$ (AS 1.4) and *address* $(i, \ldots)$ (AS 1.3) both use *checkset* to operate on the factor indicators i and j. The call by name in *checkset* implies that the formal parameters i and j

of *scan* and *address* cannot be written integers or arithmetic expressions. This is tiresome, as statements such as *scan* (0,0,...), and *scan* ($i+FS, j+FS$,...) cannot be used. Also, in loops, the input form of factor indicators in procedure calls will be the output form from the previous call, unless the parameters are reset before every call. A particularly dangerous side-effect is that the coding

$$j:=3+FS;$$

$$scan\ (j, j, \ldots);$$

implies a factor set of two factors for the first actual parameter, but as *checkset* removes *FS* from *j*, the second actual parameter refers to the single factor number 3.

A better approach, therefore, is to write *checkset* (*i*, *wi*, *ifault*), where *i* is an input parameter and *wi* an output parameter (or working *i*-value), calling *i* by value and *wi* by name. We then have:

```
procedure checkset (i, wi, ifault);
value i;
integer i, wi, ifault;
begin
ifault:=0;
if $i \geq FS$ then
        begin
        wi:=i-FS;
        if $wi \geq 2 \uparrow L[0]$ then ifault:=1
        end
else if $i \leq L[0]$ then $wi:=2 \uparrow (L[0]-i)$
             else begin
                 wi:=0;
                 ifault:=1
                 end
end checkset;
```

The use of the above form entails minor modifications to *scan* (AS 1.4) and *address* (AS 1.3) but not to their formal parameter lists. All that is required is to declare *wi* and *wj* as integers, and to replace all factor sets *i* and *j* by *wi* and *wj* except where they occur as actual parameters of *checkset*. Future algorithms will assume that these modifications have been made, but because all the algorithms using AS 1 are intended for communication purposes only, it does not seem worth publishing any further details. Copies of the revised version of AS 1 can be obtained from the author.

Errata

The original version of checkset has $>$ and $<$ whereas the above version has $\geq$ and $\leq$. This was an error in AS 1 arising from editorial amendments to the correct paper tape final version, where $\geq$ was punched as $>$ backspace underline. This was edited as $>$ with the intention of converting by hand to $\geqslant$, but the hand corrections got lost in the version from which the blocks were made.

I am grateful to Mr I. D. Hill for pointing out a minor error in *address* (AS 1.3). Here, in the middle of the procedure, the expression $F[n-i+1]$ occurs; n should be replaced by $L[0]$. This error was not found by the test programs because n happened to be global to the procedures being tested.