ALGORITHM 75
FACTORS
J. E. L. Peck,
University of Alberta, Calgary, Alberta, Canada

```
procedure factors (n,a,u,v,r,c);
comment This procedure finds all the rational linear factors of
  the polynomial a₀xⁿ + a₁xⁿ⁻¹ + ··· + aₙ₋₁x + aₙ, with integral
  coefficients. An absolute value procedure abs is assumed;
value n,a; integer r,n,c; integer array a,u,v;
begin comment We find whether p divides a₀, 1 ≤ p ≤ |a₀| and
  q divides aₙ, 0 ≤ q ≤ |aₙ|. If this is the case we try (px ± q);
integer p,q,a0,an;
r := 0; c := 1; comment r will be the number of linear factors
  and c the common constant factor;
TRY AGAIN: a0 := a[0]; an := a[n];
for p := 1 step 1 until abs(a0) do
  begin if (a0 ÷ p) × p = a0 then
    begin comment p divides a₀;
    for q := 0 step 1 until abs(an) do
      begin if q = 0 ∨ (an ÷ q) × q = an then
        begin comment q divides aₙ (or q = 0). If p = q we
          may have a common constant factor, therefore; if q
          > 1 ∧ p = 1 then
          begin integer j;
          for j := 1 step 1 until n-1 do
            if (a[j] ÷ q) × q ≠ a[j] then go to NO CONSTANT;
          for j := 0 step 1 until n do
            a[j] := a[j]/q;
          c := c × q; go to TRY AGAIN
          end the search for a common constant factor;
        NO CONSTANT:
          begin comment try (px - q) as a factor;
          integer f,g,i; f := a0; g := 1;
          comment we try x = q/p;
          for i := 1 step 1 until n do
            begin g := g × p; f := f × q + a[i] × g
            end evaluation;
          if f = 0 then
            begin comment we have found the factor (px - q);
            r := r + 1; u[r] := p; v[r] := q;
            comment there are now r linear factors;
              begin comment we divide by (px - q);
              integer i,t; t := 0;
              for i := 0 step 1 until n do
                begin a[i] := t := (a[i] + t)/p; t := t × q
                end i;
              n := n - 1
              end reduction of polynomial. Therefore;
            go to if n = 0 then REDUCED else TRY AGAIN
            end discovery of px - q as a factor. But
          if we got this far it was not a factor so try px + q;
          q := -q; if q < 0 then go to NO CONSTANT
          end trial of px ± q,
        end q divides aₙ and
      end of q loop.
    end p divides a₀, also
  end p loop, which means;
REDUCED: if n = 0 then
  begin c := c × a0; a0 := 1
  end if n = 0
end factors procedure. There are now r (r > 0) rational linear
  factors (uᵢx - vᵢ), 1 < i < r, and the reduced polynomial of
  reduced degree n replaces the original. The common constant
  factor is c. Acknowledgments to Clay Perry.
```

CERTIFICATION OF ALGORITHM 75
FACTORS [J. E. L. Peck, *Comm. ACM 5* (Jan. 1962)]
A. P. Relph
The English Electric Co., Whetstone, England

Algorithm 75 was translated using the Deuce Algol compiler and gave satisfactory results after the following corrections had been made:

```
begin if q=0∨(an÷q)×q=an then
  begin if q>1∧p=1 then
```

was changed to

```
begin if q≦1 then go to NO CONSTANT;
  if (an÷q)×q=an then
  begin if p=q then
```

```
begin c := c×a0; a0 := 1
end
```

was changed to

```
begin c := c×a[0]; a[0] := 1;
end
```

There are now r (r>0) rational linear factors ($u_i x - v_i$), $1<i<r$,

was changed to

If r>0 there are now r rational linear factors ($u_i x - v_i$), $1 \leqq i \leqq r$,

CERTIFICATION OF ALGORITHM 75
FACTORS [J. E. L. Peck, *Comm. ACM*, Jan. 1962]
J. S. Hillmore
Elliott Bros. (London) Ltd., Borehamwood, Herts., England

The following changes had to be made to the algorithm:

(1) *For* **if** $q > 1 \wedge p = 1$ **then**
put **if** $q > 1 \wedge p = q$ **then**
(2) *For* **begin** $c := c \times a0$; $a0 := 1$ **end**
put **begin** $c := c \times a[0]$; $a[0] := 1$ **end**
(3) *For* **if** $q = 0 \vee (an \div q) \times q = an$ **then**
put **if** (**if** $q = 0$ **then true else** $(an \div q) \times q = an$) **then**

This change is necessary to ensure that the term $(an \div q)$ is not evaluated when $q = 0$.

The algorithm, thus modified, was successfully run using the Elliott Algol translator on the National-Elliott 803.

To return to the state ($p=1$, $q=0$) after every factor or constant is found is inefficient. This can be avoided by substituting a[0] and a[n] for the identifiers a0 and an respectively. The procedure then becomes:

```
procedure factors (n, a, u, v, r, c);  value n, a;
  integer array a, u, v;
integer r, n, c;
begin integer p, q;
  r := 0;  c := 1;
ZERO:  if a[n]=0 then
         begin r := r+1;  u[r] := 1;  v[r] := 0;  n := n-1;
           go to ZERO
         end;
         for p := 1 step 1 until abs (a[0]) do
         begin if (a[0]÷p)×p=a[0] then
           begin for q := 1 step 1 until abs (a[n]) do
             begin if q=1 then go to NO CONSTANT;
           TRY AGAIN:  if (a[n]÷q)×q=a[n] then
                       begin integer j;
                         for j := 0 step 1 until n-1 do
                           if (a[j]÷q)×q≠a[j] then go to
                           NO CONSTANT;
                         for j := 0 step 1 until n do
                           a[j] := a[j]/q;
                         c := c×q;  go to TRY AGAIN
                       end;
        NO CONSTANT:   begin integer f, g, i;  f := a[0];
                         g := 1;
                         for i := 1 step 1 until n do
                         begin g := g×p;
                           f := f×q+a[i]×g
                         end;
                         if f=0 then
                         begin r := r+1;  u[r] := p;
                           v[r] := q;
                           begin integer i, t;  t := 0;
                             for i := 0 step 1 until n do
                             begin a[i] := t := (a[i]+t)/p;
                               t := t×q
                             end;
                             n := n-1
                           end
                           go to if n=0 then REDUCED
                             else NO CONSTANT
                         end;
                         q := -q;  if q<0 then go to NO
                           CONSTANT
                       end
             end
           end
        end;
        REDUCED:  if n=0 then
          begin c := c×a[0];  a[0] := 1
          end
end
```