```
algol<
begin
   real pi,EPS,EPS4;
   integer outputCode,oldrand;
    procedure Pmult(A1, deg1, A0, deg0, a);
    value deg0, a;
    integer deg1, deg0;
    real a;
    array A1, A0;
    begin
       comment P1(x) = P0(x)×(x-a);
       integer i;
       deg1 := deg0+1;
       for i:=0 step 1 until deg0 do
       A1[i+1]:=A0[i];
       A1[0]:=0;
       for i:=0 step 1 until deg0 do
       A1[i] := A1[i]-a×A0[i]
   end Pmult;
   real procedure RANDOM;
   code RANDOM,oldrand;
   1, 37;
   2, 44;
   pm a2,mln re1     ; M:=oldrand; RM:=oldrand×16807;
   dl re2,gm a2      ; M:=RM mod 2147483647;
   xr   , nkf 39     ; RF:=M;
   dkf re3, srf c44  ; RF:=M/2147483647.0-0.5;
   grf p-1, hv r4
e1: qq 16807.39
e2: qq 2147483647.39
f
e3: 2147483647.0
    e;
   real procedure CubeRoot(x);
   value x;
   real x;
   CubeRoot:=(if x<0 then
      -exp(ln(-x)/3)
      else if x>0 then
      exp(ln(x)/3)
      else
      0);
   real procedure atan2(y,x);
   value y,x;
   real y,x;
    atan2:=(
      if x>0 then
         arctan(y/x)
      else if x<0 then
         arctan(y/x)+(if y≥0 then pi else -pi)
      else
         (if y>0 then pi/2 else -pi/2));
   procedure LinearRoot(a, z);
   array a;
   real z;
   begin
      if a[2]=0 then
        z:=0
      else
        z:=-a[1]/a[2]
   end LinearRoot;
   procedure OneLargeTwoSmall(a1,a2,a4,w,z);
   value a1,a2,a4,w;
   real a1,a2,a4,w;
```

```
array z;
begin
   array aq[1:3];
   aq[1] := a1;
   aq[2] := a2+a1/w;
   aq[3] := -a4×w;
   QuadraticRoots(aq, z);
   z[3,1] := w;
   z[3,2] := 0;
   if z[1,2] ≠0 then
   begin
      z[3,1]:=z[2,1]; z[3,2]:=z[2,2];
      z[2,1]:=z[1,1]; z[2,2]:=z[1,2];
      z[1,1]:=w;
      z[1,2]:=0
   end
end OneLargeTwoSmall;
procedure QuadraticRoots(a, z);
array a,z;
begin
   real d, r, w, x, y;
   if a[1]=0 then
   begin
      z[1,1]:=0;
      z[1,2]:=0;
      z[2,1]:=-a[2]/a[3];
      z[2,2]:=0;
      outputCode:=21;
      goto end
   end;
   d:=a[2]↑2 - 4×a[1]×a[3];
   if abs(d) ≤ 2×EPS×a[2]↑2 then
   begin
      z[1,1] := -0.5×a[2]/a[3];
      z[1,2] := 0;
      z[2,1] := z[1,1];
      z[2,2] := z[1,2];
      outputCode := 22;
      goto end
   end;
   r := sqrt(abs(d));
   if d<0 then
   begin
      x := -0.5×a[2]/a[3];
      y := abs(0.5×r/a[3]);
      z[1,1] := x;
      z[1,2] := y;
      z[2,1] := x;
      z[2,2] := -y;
      outputCode := 23;
      goto end
   end;
   if a[2] ≠ 0 then
   begin
      w := -(a[2]+sign(a[2])×abs(r));
      z[1,1] := 2×a[1]/w;
      z[1,2] := 0;
      z[2,1] := 0.5×w/a[3];
      z[2,2] := 0;
      outputCode := 22;
      goto end
   end;
   x := abs(0.5×r/a[3]);
   z[1,1] := x;
```

```
      z[1,2] := 0;
      z[2,1] := -x;
      z[2,2] := 0;
      outputCode := 22;
end:
   end QuadraticRoots;
   procedure CubicRoots(a, z);
   array a, z;
   begin
      comment
         PURPOSE - Compute the roots of the real polynomial
             a[1] + a[2]×z + ... + a[4]×z↑3

         Note: It is assumed that a[4] is non-zero. No test is made here
      ;
      real rt3,arg, c, cf, d, p, p1, q, q1, r, ra, rb, rq, rt, r1, s, sf, sq, sum, t,
      array aq[1:3];
      rt3 := sqrt(3);
      if a[1]=0 then
      begin
         comment One root is obviously zero;
         array a2[1:3], z2[1:2,1:2];
         a2[1] := a[2];
         a2[2] := a[3];
         a2[3] := a[4];
         comment remaining 2 roots here;
         QuadraticRoots(a2, z2);
         z[1,1] := 0;
         z[1,2] := 0;
         z[2,1] := z2[1,1];
         z[2,2] := z2[1,2];
         z[3,1] := z2[2,1];
         z[3,2] := z2[2,2];
         goto end
      end;

      p:=a[3]/(3×a[4]);
      q:=a[2]/a[4];
      r:=a[1]/a[4];
      tol:=EPS4;

      c:=0;
      t:=a[2]-p×a[3];
      if abs(t)>tol×abs(a[2]) then c:=t/a[4];

      t:=2×p↑2 - q;
      if abs(t) ≤ tol×abs(q) then t := 0;
      d := r + p×t;
      if abs(t) ≤ tol×abs(r) then goto l110;

      comment sq = (a[4]/s)↑2 × (c↑3/27 + d↑2/4) ;

      s := abs(a[1]);
      if abs(a[2])>s then s:=abs(a[2]);
      if abs(a[3])>s then s:=abs(a[3]);
      p1 := a[3]/(3×s);
      q1 := a[2]/s;
      r1 := a[1]/s;

      t1 := q- 2.25×p↑2;
      if abs(t1) ≤ tol×abs(q) then t1 := 0;
      w := 0.25×r1↑2;
      w1 := 0.5×p1×r1×t;
      w2 := q1↑2×t1/27;
```

```
      if w1 ≥ 0 then
      begin
         w := w+w1;
         sq := w+w2
      end
      else if w2 < 0 then
         sq := w + (w1+w2)
      else
      begin
         w := w+w2;
         sq := w+w1
      end;

      if abs(sq) ≤ tol×w then sq := 0;
      rq := abs(s/a[4])×sqrt(abs(sq));
      if sq ≥ 0 then goto l40;

      comment all roots are real;

      arg := atan2(rq, -0.5×d);
      cf := cos(arg/3);
      sf := sin(arg/3);
      rt := sqrt(-c/3);
      y1 := 2×rt×cf;
      y2 := -rt×(cf+ rt3×sf);
      y3 := -(d/y1)/y2;

      x1 := y1 - p;
      x2 := y2 - p;
      x3 := y3 - p;

      if abs(x1)>abs(x2) then Swap(x1,x2);
      if abs(x2)>abs(x3) then Swap(x2,x3);
      if abs(x1)>abs(x2) then Swap(x1,x2);

      w := x3;

      if abs(x2) < 0.1×abs(x3) then goto l70;
      if abs(x1) < 0.1×abs(x2) then x1 := -(r/x3)/x2;
      z[1,1] := x1;
      z[1,2] := 0;
      z[2,1] := x2;
      z[2,2] := 0;
      z[3,1] := x3;
      z[3,2] := 0;
      outputCode := 33;
      goto end;
      comment real and complex roots;
l40:  ra := CubeRoot(-0.5×d - abs(rq)×sign(d));
      rb := -c/(3×ra);
      t := ra + rb;
      w := -p;
      x := -p;
      if abs(t) ≤ tol×abs(ra) then goto  l41;
      w := t - p;
      x := -0.5×t - p;
      if abs(x) ≤ tol×abs(p) then x:=0;
l41:  t := abs(ra - rb);
      y := 0.5×rt3×t;

      if t ≤ tol×abs(ra) then goto l60;
      if abs(x)<abs(y) then goto l50;
      s := abs(x);
```

```
      t := y/x;
      goto l51;
l50:  s := abs(y);
      t := x/y;
l51:  if s < 0.1×abs(w) then goto l70;
      w1 := w/s;
      sum := 1 + t↑2;
      if w1↑2 < 0.01×sum then w := -((r/sum)/s)/s;
      z[1,1] := w;
      z[1,2] := 0;
      z[2,1] := x;
      z[2,2] := y;
      z[3,1] := x;
      z[3,2] := -y;
      outputCode := 32;
      goto end;
      comment At least two roots are equal;
l60:  if abs(x)<abs(w) then goto l61;
      if abs(w)<0.1×abs(x) then w := -(r/x)/x;
      z[1,1] := w;
      z[1,2] := 0;
      z[2,1] := x;
      z[2,2] := 0;
      z[3,1] := x;
      z[3,2] := 0;
      outputCode := 31;
      goto end;
l61:  if abs(x) < 0.1×abs(w) then goto l70;
      z[1,1] := x;
      z[1,2] := 0;
      z[2,1] := x;
      z[2,2] := 0;
      z[3,1] := w;
      z[3,2] := 0;
      outputCode := 31;
      goto end;
      comment
         Here w is much larger in magnitude than the other roots.
         As a result, the other roots may be exceedingly inaccurate
         because of roundoff errors. To deal with this, a quadratic
         is formed whose roots are the same as the smaller roots of
         the cubic.  This quadratic is then solved.
      ;

l70:  aq[1] := a[1];
      aq[2] := a[2] + a[1]/w;
      aq[3] := -a[4]×w;
      QuadraticRoots(aq, z);
      z[3,1] := w;
      z[3,2] := 0;
      outputCode := 31;
      if z[1,2]=0 then goto end;
      z[3,1] := z[2,1];
      z[3,2] := z[2,2];
      z[2,1] := z[1,1];
      z[2,2] := z[1,2];
      z[1,1] := w;
      z[1,2] := 0;
      outputCode := 32;
      goto end;
      comment Case when d = 0;
l110: z[1,1] := -p;
      z[1,2] := 0;
      w := sqrt(abs(c));
```

```
      if c<0 then goto l120;
      z[2,1] := -p;
      z[2,2] := w;
      z[3,1] := -p;
      z[3,2] := -w;
      outputCode := 32;
      goto end;
l120: if p ≠ 0 then goto l130;
      z[2,1] := w;
      z[2,2] := 0;
      z[3,1] := -w;
      z[3,2] := 0;
      outputCode := 31;
      goto end;
l130: x := -(p+abs(w)×sign(p));
      z[3,1] := x;
      z[3,2] := 0;
      t := 3×a[1]/(a[3]×x);
      if abs(p)>abs(t) then goto l131;
      z[2,1] := t;
      z[2,2] := 0;
      outputCode := 31;
      goto end;
l131: z[2,1] := z[1,1];
      z[2,2] := z[1,2];
      z[1,1] := t;
      z[1,2] := 0;
      outputCode := 31;
end:
   end CubicRoots;
   procedure QuarticRoots(a,z);
   array a,z;
   begin
      comment
         PURPOSE - Compute the roots of the real polynomial
             a[1] + a[2]×z + ... + a[5]×z↑4

         Note: It is assumed that a[5] is non-zero. No test is made here
      ;
      real wr,wi,b,b2, c, d, e, h, p, q, r, t;
      array temp[1:4];
      real u, v, v1, v2, x, x1, x2, x3, y;
      DEBUGr(⎨<a[1]⎬, a[1]);
      if a[1]=0 then
      begin
         array a3[1:4],z3[1:4,1:2];
         integer i;
         comment One root is obviously zero;
         z[1,1] := 0;
         z[1,2] := 0;
         for i:=1 step 1 until 4 do a3[i]:=a[i+1];
         CubicRoots(a3,z3);
         for i:=1 step 1 until 4 do
         begin
            z[i+1,1] := z3[i,1];
            z[i+1,2] := z3[i,2]
         end;
         goto end
      end;

      b := a[4]/(4×a[5]);
      c := a[3]/a[5];
      d := a[2]/a[5];
      e := a[1]/a[5];
```

```
      b2 := b↑2;

      p := 0.5×(c - 6×b2);
      q := d - 2×b×(c - 4×b2);
      r := b2×(c - 3×b2) - b×d + e;

      comment

         Solve the resolvent cubic equation. The cubic has at least one
         nonnegative real root. If w1, w2, w3 are the roots of the cubic
         then the roots of the original equation are

            z = -b + csqrt(w1) + csqrt(w2) + csqrt(w3)

         where the signs of the square roots are chosen so
         that csqrt(w1) × csqrt(w2) × csqrt(w3) = -q/8
      ;

      temp[1] := -q×q/64;
      temp[2] := 0.25×(p×p - r);
      temp[3] := p;
      temp[4] := 1;
      DEBUGr(╟<<temp[1]╢, temp[1]);
      DEBUGr(╟<<temp[2]╢, temp[2]);
      DEBUGr(╟<<temp[3]╢, temp[3]);
      DEBUGr(╟<<temp[4]╢, temp[4]);
      CubicRoots(temp,z);
      DEBUGr(╟<<z[1,1]╢, z[1,1]);
      DEBUGr(╟<<z[1,2]╢, z[1,2]);
      DEBUGr(╟<<z[2,1]╢, z[2,1]);
      DEBUGr(╟<<z[2,2]╢, z[2,2]);
      DEBUGr(╟<<z[3,1]╢, z[3,1]);
      DEBUGr(╟<<z[3,2]╢, z[3,2]);
      if z[2,2]≠0 then goto l60;
      comment
         The resolvent cubic has only real roots
         Reorder the roots in increasing order
      ;
      x1 := z[1,1];
      x2 := z[2,1];
      x3 := z[3,1];
      if x1>x2 then Swap(x1,x2);
      if x2>x3 then Swap(x2,x3);
      if x1>x2 then Swap(x1,x2);

      u := 0;
      if x3 > 0 then u:=sqrt(x3);
      if x2 ≤ 0 then goto l41;
      if x1 ≥ 0 then goto l30;
      if abs(x1) > x2 then goto l40;
      x1 := 0;
l30:  x1 := sqrt(x1);
      x2 := sqrt(x2);
      if q>0 then x1 := -x1;
      temp[1] := (( x1 + x2) + u) - b;
      temp[2] := ((-x1 - x2) + u) - b;
      temp[3] := (( x1 - x2) - u) - b;
      temp[4] := ((-x1 + x2) - u) - b;
      DEBUGr(╟<<temp[1] before sort╢, temp[1]);
      DEBUGr(╟<<temp[2] before sort╢, temp[2]);
      DEBUGr(╟<<temp[3] before sort╢, temp[3]);
      DEBUGr(╟<<temp[4] before sort╢, temp[4]);
      SelectSort(4,temp);
      DEBUGr(╟<<temp[1] after sort╢, temp[1]);
```

```
      DEBUGr('<temp[2] after sort>', temp[2]);
      DEBUGr('<temp[3] after sort>', temp[3]);
      DEBUGr('<temp[4] after sort>', temp[4]);
      if abs(temp[1]) ≥ 0.1×abs(temp[4]) then goto l31;
      t := temp[2]×temp[3]×temp[4];
      if t ≠ 0 then temp[1] := e/t;
l31:  z[1,1] := temp[1];
      z[1,2] := 0;
      z[2,1] := temp[2];
      z[2,2] := 0;
      z[3,1] := temp[3];
      z[3,2] := 0;
      z[4,1] := temp[4];
      z[4,2] := 0;
      outputCode := 40;
      goto end;

l40:  v1 := sqrt(abs(x1));
      v2 := 0;
      goto l50;
l41:  v1 := sqrt(abs(x1));
      v2 := sqrt(abs(x2));
      if q<0 then u := -u;
l50:  x := -u - b;
      y := v1 - v2;
      z[1,1] := x;
      z[1,2] := y;
      z[2,1] := x;
      z[2,2] := -y;
      x := u - b;
      y := v1 + v2;
      z[3,1] := x;
      z[3,2] := y;
      z[4,1] := x;
      z[4,2] := -y;
      outputCode := 44;
      goto end;
      comment
         The resolvent cubic has complex roots
      ;
l60:  t := z[1,1];
      x := 0;
      goto if t<0 then l61 else if t=0 then l70 else l62;
l61:  h := abs(z[2,1]) + abs(z[2,2]);
      if abs(t) ≤ h then goto l70;
      goto l80;
l62:  x := sqrt(t);
      if q>0 then x := -x;
l70:  csqrt(z[2,1],z[2,2],wr,wi);
      u := 2×wr;
      v := 2×abs(wi);
      t := x - b;
      x1 := t + u;
      x2 := t - u;
      if abs(x1) ≤ abs(x2) then goto l71;
      t := x1;
      x1 := x2;
      x2 := t;
l71:  u := -x - b;
      h := u↑2 + v↑2;
      if x1↑2 < 0.01×MIN(x2↑2,h) then x1 := e/(x2×h);
      z[1,1] := x1;
      z[1,2] := 0;
      z[2,1] := x2;
```

```
      z[2,2] := 0;
      z[3,1] := u;
      z[3,2] := v;
      z[4,1] := u;
      z[4,2] := -v;
      outputCode := 42;
      goto end;
l80:  v := sqrt(abs(t));
      z[1,1] := z[3,1] := -b;
      z[1,2] := z[3,2] := v;
      z[2,1] := z[4,1] := -b;
      z[2,2] := z[4,2] := -v;
      outputCode := 44;
end:
   end QuarticRoots;
   procedure SelectSort(n,a);
   value n;
   integer n;
   array a;
   begin
      integer i,j;
      for i:=1 step 1 until n-1 do
      for j:=i+1 step 1 until n do
      if a[i]>a[j] then Swap(a[i],a[j])
   end SelectSort;
   procedure DEBUGr(text, r);
   value r;
   string text;
   real r;
   begin
      if kbon  then
      begin
         writecr;
         writetext(text);
         writetext({<:  });
         write({-d.dddddd10-dd},r)
      end
   end;
   integer procedure SolvePolynomial(quarticCoeff, cubicCoeff, quadraticCoeff,
       linearCoeff, constantCoeff, root1r, root1i, root2r, root2i, root3r, root3i,
       root4r, root4i);
   value quarticCoeff, cubicCoeff, quadraticCoeff, linearCoeff, constantCoeff;
   real quarticCoeff, cubicCoeff, quadraticCoeff, linearCoeff, constantCoeff,
      root1r, root1i, root2r, root2i, root3r, root3i, root4r, root4i;
   begin
      array a[1:5],z[1:4,1:2];
      outputCode := -9999;
      a[1] := constantCoeff;
      a[2] := linearCoeff;
      a[3] := quadraticCoeff;
      a[4] := cubicCoeff;
      a[5] := quarticCoeff;
      DEBUGr({<quarticCoeff}, quarticCoeff);
      if quarticCoeff ≠ 0 then
         QuarticRoots(a,z)
      else if cubicCoeff ≠ 0 then
         CubicRoots(a,z)
      else if quadraticCoeff ≠ 0 then
         QuadraticRoots(a,z)
      else if linearCoeff ≠ 0 then
      begin
         z[1,1] := -constantCoeff/linearCoeff;
         z[1,2] := 0;
         outputCode := 1
```

```
   end
   else
      outputCode := 0;
   if outputCode>0 then
   begin
      root1r := z[1,1];
      root1i := z[1,2]
   end;
   if outputCode>1 then
   begin
      root2r := z[2,1];
      root2i := z[2,2]
   end;
   if outputCode>23 then
   begin
      root3r := z[3,1];
      root3i := z[3,2]
   end;
   if outputCode>39 then
   begin
      root4r := z[4,1];
      root4i := z[4,2]
   end;
   SolvePolynomial := outputCode
end SolvePolynomial;
real procedure cabs(r,i);
value r,i;
real r,i;
cabs:=sqrt(r↑2+i↑2);
procedure csqrt(xr,xi,zr,zi);
value xr,xi;
real xr,xi,zr,zi;
begin
   real r,sr,rzr;
   r := cabs(xr,xi);
   sr := sqrt(r);
   zr:=xr+r;
   zi:=xi;
   rzr := cabs(zr,zi);
   zr:=zr×sr/rzr;
   zi:=zi×sr/rzr;
end csqrt;
procedure Swap(a,b);
real a,b;
begin
   real t;
   t:=a;
   a:=b;
   b:=t
end Swap;
real procedure MIN(a,b);
value a,b;
real a,b;
MIN := if a<b then a else b;
array coef[0:4],root[1:4,1:2];
integer deg1, deg2, deg3, deg4;
real r1,r2,r3,r4;
array A1[0:1],A2[0:2],A3[0:3],A4[0:4];
array sort1,sort2[1:4];
integer code,i;
EPS := real 10 996 10 256 10 0 10 0;
EPS4 := real 10 998 10 256 10 0 10 0;
pi := 3.1415926536;
oldrand := 100001;
```

```
   select(16);
again:
   coef[0] := RANDOM×10;
   coef[1] := -RANDOM;
   coef[2] := RANDOM×1₁₀-2;
   coef[3] := -RANDOM×1₁₀-4;
   coef[4] := RANDOM×1₁₀-6;
   r1 := RANDOM×10;
   r2 := RANDOM×10;
   r3 := RANDOM×10;
   r4 := RANDOM×10;
   sort1[1] := r1;
   sort1[2] := r2;
   sort1[3] := r3;
   sort1[4] := r4;
   A1[0] := r1;
   A1[1] := -1;
   deg1 := 1;
   Pmult(A2, deg2, A1, deg1, r2);
   Pmult(A3, deg3, A2, deg2, r3);
   Pmult(A4, deg4, A3, deg3, r4);
   writecr; write(|<-d.dddddd₁₀-dd>|,r1);
   writecr; write(|<-d.dddddd₁₀-dd>|,r2);
   writecr; write(|<-d.dddddd₁₀-dd>|,r3);
   writecr; write(|<-d.dddddd₁₀-dd>|,r4);

   for i:=0 step 1 until 4 do
   begin
      coef[i] := A4[i];
      writecr;
      write(|<dd>|, i);
      write(|<  -d.dddddd₁₀-dd>|, coef[i])
   end;
   for i:=1 step 1 until 4 do
   begin
      root[i,1] := 1234;
      root[i,2] := 5678;
   end;
   code := SolvePolynomial(coef[4],coef[3],coef[2],coef[1],coef[0],
           root[1,1], root[1,2], root[2,1], root[2,2],
           root[3,1], root[3,2], root[4,1], root[4,2]);
   writecr;
   writetext(if code=0 then |<degenerate equation>|
     else if code=1 then |<one real root>|
     else if code=21 then |<two identical real roots>|
     else if code=22 then |<two distinct real roots>|
     else if code=23 then |<two complex roots>|
     else if code=31 then |<multiple real roots>|
     else if code=32 then |<one real and two complex roots>|
     else if code=33 then |<three distinct real roots>|
     else if code=40 then |<four real roots>|
     else if code=42 then |<two real and two complex roots>|
     else if code=44 then |<four complex roots>|
     else if code=-9999 then |<unassigned code>|
     else |<unknown code>|);
   for i:=1 step 1 until 4 do sort2[i] := root[i,1];
   SelectSort(4,sort1);
   SelectSort(4,sort2);
   for i:=1 step 1 until 4 do
   begin
      writecr;
      write(|<  -d.dddddd₁₀-dd>|, sort1[i], sort2[i], sort1[i]-sort2[i]);
   end;
   for i:=0 step 1 until 3 do
```

```
   begin
      coef[i] := A4[i+1]×(i+1);
      writecr;
      write(‹dd›, i);
      write(‹  -d.dddddd10-dd›, coef[i])
   end;
   for i:=1 step 1 until 4 do
   begin
      root[i,1] := 1234;
      root[i,2] := 5678;
   end;
   code := SolvePolynomial(0,coef[3],coef[2],coef[1],coef[0],
           root[1,1], root[1,2], root[2,1], root[2,2],
           root[3,1], root[3,2], root[4,1], root[4,2]);
   writecr;
   writetext(if code=0 then ‹<degenerate equation›
     else if code=1 then ‹<one real root›
     else if code=21 then ‹<two identical real roots›
     else if code=22 then ‹<two distinct real roots›
     else if code=23 then ‹<two complex roots›
     else if code=31 then ‹<multiple real roots›
     else if code=32 then ‹<one real and two complex roots›
     else if code=33 then ‹<three distinct real roots›
     else if code=40 then ‹<four real roots›
     else if code=42 then ‹<two real and two complex roots›
     else if code=44 then ‹<four complex roots›
     else if code=-9999 then ‹<unassigned code›
     else ‹<unknown code›);
   for i:=1 step 1 until 4 do
   begin
      writecr;
      write(‹  -d.dddddd10-dd›, root[i,1], root[i,2]);
   end;
   goto again;
bad:
end;
t<
```