>
Here we look to break down an element of Q $((-2)^{\frac{1}{3}})$ *intoasetofprimeswithrelativelysmallnorms.Weneedprocedu*
Norm, Multiplication, Inversion and Division in the field
The three roots of the equation z ^{3} = -2 are given below. If z1 is the real root then the two conjugate complex roots are z2 = wz and z3 = w^2z where
> $solve(z^3 = -2, z);$

$$-2^{\frac{1}{3}}, \frac{2^{\frac{1}{3}}}{2} - \frac{\mathrm{I}\sqrt{3}\,2^{\frac{1}{3}}}{2}, \frac{2^{\frac{1}{3}}}{2} + \frac{\mathrm{I}\sqrt{3}\,2^{\frac{1}{3}}}{2} \tag{1}$$

> $solve(w^2 + w + 1, w);$

$$-\frac{1}{2} + \frac{\mathrm{I}\sqrt{3}}{2}, -\frac{1}{2} - \frac{\mathrm{I}\sqrt{3}}{2} \tag{2}$$

Each element of the ring has the form a + bz +czˆ2 where a is an integer which we write as [a,b,c]. The norm of an element [a,b,c] is (a + bz + czˆ2)(a+bwz+cwˆ2)(a+bwˆ2z+cwzˆ2). Because zˆ3 = -2 and wˆ2+w+1 = 0 this simplifies as norm1 below.

> $norm1 := \mathrm{proc}(a, b, c)\,\mathrm{global}\,k;\ k := a^3 - 2 \cdot b^3 + 4 \cdot c^3 + 6 \cdot a \cdot b \cdot c; \mathrm{end};$

$$norm1 := \boldsymbol{proc}\,(a, b, c) \quad \boldsymbol{global} \quad k; \quad k := a^3 - 2 * b^3 + 4 * c^3 + 6 * b * a * c \quad \text{end proc} \tag{3}$$

> $norm1\,(66, 53, 0);$

$$-10258 \tag{4}$$

The following procedures perform multiplication, inversion and division in our field

> $mult2 := \mathrm{proc}(a, b, c, d, e, f)\mathrm{global}\,mul1, mul2, mul3;\ mul1 := a \cdot d - 2 \cdot b \cdot f - 2 \cdot c \cdot e;$
$mul2 := a \cdot e + b \cdot d - 2 \cdot c \cdot f;$
$mul3 := a \cdot f + b \cdot e + c \cdot d;$
$RETURN(mul1, mul2, mul3); \mathrm{end};$

$$mult2 := \boldsymbol{proc}\,(a, b, c, d, e, f) \quad \boldsymbol{global} \quad mul1, mul2, mul3; \quad mul1 := a * d - 2 * b * f - 2 * c * e; \quad mul2 := a * e + b * d - 2 * c * f; \quad mul3 := a * f + b * e + c * d; \quad RETURN(mul1, mul2, mul3) \quad \text{end proc} \tag{5}$$

> $invert2 := \mathrm{proc}(a, b, c)\,\mathrm{global}\,inv1, inv2, inv3;\ inv1 := \frac{(a^2 + 2 \cdot b \cdot c)}{norm1\,(a, b, c)};\ inv2 := \frac{(-a \cdot b - 2 \cdot c^2)}{norm1\,(a, b, c)};$
$inv3 := \frac{(b^2 - a \cdot c)}{norm1\,(a, b, c)};\ RETURN(inv1, inv2, inv3); \mathrm{end};$

$$invert2 := \boldsymbol{proc}\,(a, b, c) \quad \boldsymbol{global} \quad inv1\,, inv2, inv3; \quad inv1 := \left(a^2 + 2*b*c\right)/norm1(a, b, c)\,; \quad inv2 := \left(-b*a - 2*c^2\right)/norm1(a, b, c)\,; \quad inv3 := \left(b^2 - a*c\right)/norm1(a, b, c)\,; \quad RETURN(inv1\,, inv2, inv3) \quad \text{end proc} \tag{6}$$

> $divide3 := \mathrm{proc}(a, b, c, d, e, f); mult2(a, b, c, invert2(d, e, f)); \mathrm{end};$

$$divide3 := \boldsymbol{proc}\,(a, b, c, d, e, f) \quad mult2(a, b, c, invert2(d, e, f)) \quad \text{end proc} \tag{7}$$

>

Divide3 is a procedure to produce the result of dividing two triples of the form (a,b,c) = a +bz + cz $^{2.0}$If this division produces an element with integer values we say (a,b,c) is divisible. Next we define our factor base:

> $U := [1, 1, 0]; A := [0, 1, 0]; B := [-1, 1, 0]; C := [1, 0, 1]; D1 := [1, 1, -1]; E := [1, -2, 0]; F := [3, 0, -1];$

$$\begin{aligned} U &:= [1, 1, 0] \\ A &:= [0, 1, 0] \\ B &:= [-1, 1, 0] \\ C &:= [1, 0, 1] \\ D1 &:= [1, 1, -1] \\ E &:= [1, -2, 0] \\ F &:= [3, 0, -1] \end{aligned} \tag{8}$$

If A = [0,1,0], B = [-1,-1,0], C = [1,0,1], D = [1,1,-1], E = [1,-2,0], F = [3,0,-1] then we have prime elements with norm = +/- 2, 3, 5, 11, 17, 23

We need procedures to decide on divisibility and then how many times we can divide out the prime element. We also need to find how many times we can divide out by a unit element which we choose as U = [1,1,0]

The 'divisibleby' procedures check if the outcome of the division rule produces integer values in each of the three positions s[1], s[2], and s[3]. If it does we can perform the 'divideby' procedure to extract all powers of the member of the factor base.

> $divisiblebyA := \mathrm{proc}(a, b, c);\ s := [mult2(a, b, c, invert2(0, 1, 0))];$ if $type(s[1], integer) \wedge type(s[2], integer) \wedge type(s[3], integer)$ then $true$ else $false$; fi; end; $divisiblebyB := \mathrm{proc}(a, b, c);\ s := [mult2(a, b, c, invert2(-1, 1, 0))];$ if $type(s[1], integer) \wedge type(s[2], integer) \wedge type(s[3], integer)$ then $true$ else $false$; fi; end; $divisiblebyC := \mathrm{proc}(a, b, c);\ s :=$

$[mult2(a, b, c, invert2(1, 0, 1))]$; if $type(s[1], integer) \wedge type(s[2], integer) \wedge type(s[3], integer)$ then $true$ else $false$; f
proc(a, b, c); $s := [mult2(a, b, c, invert2(1, 1, -1))]$; if $type(s[1], integer) \wedge type(s[2], integer) \wedge$
$type(s[3], integer)$ then $true$ else $false$; fi; end; $divisiblebyE$:= proc(a, b, c); s :=
$[mult2(a, b, c, invert2(1, -2, 0))]$; if $type(s[1], integer) \wedge type(s[2], integer) \wedge type(s[3], integer)$ then $true$ else $false$
proc(a, b, c); $s := [mult2(a, b, c, invert2(3, 0, -1))]$; if $type(s[1], integer) \wedge type(s[2], integer) \wedge$
$type(s[3], integer)$ then $true$ else $false$; fi; end; $divisiblebyU$:= proc(a, b, c); s :=
$[mult2(a, b, c, invert2(1, 1, 0))]$; if $type(s[1], integer) \wedge type(s[2], integer) \wedge type(s[3], integer)$ then $true$ else $false$; fi

Warning, (in divisiblebyA) ‘s‘ is implicitly declared localWarning, (in divisiblebyB) ‘s‘ is implicitly declared localWarning, (in divisiblebyC) ‘s‘ is implicitly declared localWarning, (in divisiblebyD) ‘s‘ is implicitly declared localWarning, (in divisiblebyE) ‘s‘ is implicitly declared localWarning, (in divisiblebyF) ‘s‘ is implicitly declared localWarning, (in divisiblebyU) ‘s‘ is implicitly declared local

$divisiblebyA :=$
proc (a, b, c) **local** s;
$s := [mult2(a, b, c, invert2(0, 1, 0))]$; **if**
$type(s[1], integer) \wedge type(s[2], integer) \wedge$
$type(s[3], integer)$
then $true$
else $false$ end if
end proc

$divisiblebyB :=$
proc (a, b, c) **local** s;
$s := [mult2(a, b, c, invert2(-1, 1, 0))]$; **if**
$type(s[1], integer) \wedge type(s[2], integer) \wedge$
$type(s[3], integer)$
then $true$
else $false$ end if
end proc

$divisiblebyC :=$
proc (a, b, c) **local** s;
$s := [mult2(a, b, c, invert2(1, 0, 1))]$; **if**
$type(s[1], integer) \wedge type(s[2], integer) \wedge$
$type(s[3], integer)$
then $true$
else $false$ end if
end proc

$$
\begin{aligned}
&\mathit{divisiblebyD} \coloneqq \\
&\quad \boldsymbol{proc}(a,b,c) \quad \boldsymbol{local} \quad s; \\
&\quad s \coloneqq [\mathit{mult2}(a,b,c,\mathit{invert2}(1,1,-1))]; \quad \boldsymbol{if} \\
&\quad \mathit{type}(s[1],\mathit{integer}) \quad \wedge \quad \mathit{type}(s[2],\mathit{integer}) \quad \wedge \\
&\quad \mathit{type}(s[3],\mathit{integer}) \\
&\quad \boldsymbol{then} \quad \mathit{true} \\
&\quad \boldsymbol{else} \quad \mathit{false} \quad \text{end if} \\
&\quad \text{end proc} \\
&\mathit{divisiblebyE} \coloneqq \\
&\quad \boldsymbol{proc}(a,b,c) \quad \boldsymbol{local} \quad s; \\
&\quad s \coloneqq [\mathit{mult2}(a,b,c,\mathit{invert2}(1,-2,0))]; \quad \boldsymbol{if} \\
&\quad \mathit{type}(s[1],\mathit{integer}) \quad \wedge \quad \mathit{type}(s[2],\mathit{integer}) \quad \wedge \\
&\quad \mathit{type}(s[3],\mathit{integer}) \\
&\quad \boldsymbol{then} \quad \mathit{true} \\
&\quad \boldsymbol{else} \quad \mathit{false} \quad \text{end if} \\
&\quad \text{end proc} \\
&\mathit{divisiblebyF} \coloneqq \\
&\quad \boldsymbol{proc}(a,b,c) \quad \boldsymbol{local} \quad s; \\
&\quad s \coloneqq [\mathit{mult2}(a,b,c,\mathit{invert2}(3,0,-1))]; \quad \boldsymbol{if} \\
&\quad \mathit{type}(s[1],\mathit{integer}) \quad \wedge \quad \mathit{type}(s[2],\mathit{integer}) \quad \wedge \\
&\quad \mathit{type}(s[3],\mathit{integer}) \\
&\quad \boldsymbol{then} \quad \mathit{true} \\
&\quad \boldsymbol{else} \quad \mathit{false} \quad \text{end if} \\
&\quad \text{end proc} \\
&\mathit{divisiblebyU} \coloneqq \\
&\quad \boldsymbol{proc}(a,b,c) \quad \boldsymbol{local} \quad s; \\
&\quad s \coloneqq [\mathit{mult2}(a,b,c,\mathit{invert2}(1,1,0))]; \quad \boldsymbol{if} \\
&\quad \mathit{type}(s[1],\mathit{integer}) \quad \wedge \quad \mathit{type}(s[2],\mathit{integer}) \quad \wedge \\
&\quad \mathit{type}(s[3],\mathit{integer}) \\
&\quad \boldsymbol{then} \quad \mathit{true} \\
&\quad \boldsymbol{else} \quad \mathit{false} \quad \text{end if} \\
&\quad \text{end proc}
\end{aligned} \tag{9}
$$

> $\mathit{dividebyA} \coloneqq$ proc(a,b,c) global $\mathit{countA}, s; t;$ $\mathit{oldk1} \coloneqq a$: $\mathit{oldk2} \coloneqq b$: $\mathit{oldk3} \coloneqq c$: $\mathit{countA} \coloneqq 0;$ $s \coloneqq [a,b,c];$ while $\mathit{divisiblebyA}(s[1],s[2],s[3]) = \mathit{true}$ do $\mathit{countA} \coloneqq \mathit{countA}+1;$ $\mathit{k1} \coloneqq s[1]; \mathit{k2} \coloneqq s[2]; \mathit{k3} \coloneqq s[3]; s \coloneqq [\mathit{divide3}(\mathit{k1},\mathit{k2},\mathit{k3},0,1,0)];$ od; $s; t \coloneqq (s[1],s[2],s[3]);$ end;

Warning, (in dividebyA) ‘oldk1‘ is implicitly declared localWarning, (in dividebyA) ‘oldk2‘ is implicitly declared localWarning, (in dividebyA) ‘oldk3‘ is implicitly declared localWarning, (in dividebyA) ‘k1‘ is implicitly declared localWarning, (in dividebyA) ‘k2‘ is implicitly declared localWarning, (in dividebyA) ‘k3‘ is implicitly declared localWarning, (in dividebyA) ‘t‘ is implicitly declared local

$dividebyA := \boldsymbol{proc}(a, b, c) \quad \boldsymbol{local} \quad oldk1, oldk2, oldk3, k1, k2, k3, t; \quad \boldsymbol{global} \; countA, s; \quad t; \quad oldk1 := a; \quad o$ (10)

$= true \quad \boldsymbol{do} \quad countA := countA + 1; \quad k1 := s[1]; \quad k2 := s[2]; \quad k3 := s[3]; \quad s := [divide3(k1, k2, k3, 0, 1$

> $dividebyB := \mathrm{proc}(a, b, c)\, \mathrm{global}\; countB, s; t;\; oldk1 := a{:}\; oldk2 := b{:}\; oldk3 := c{:}\; countB := 0;\; s := [a, b, c];\, ;\, \mathrm{while}\; divisiblebyB(s[1], s[2], s[3])\; \mathrm{do}\; countB := countB + 1;\; k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1, k2, k3, -1, 1, 0)]; \mathrm{od};\; s; t := (s[1], s[2], s[3]); \mathrm{end};$

Warning, (in dividebyB) ‘oldk1‘ is implicitly declared localWarning, (in dividebyB) ‘oldk2‘ is implicitly declared localWarning, (in dividebyB) ‘oldk3‘ is implicitly declared localWarning, (in dividebyB) ‘k1‘ is implicitly declared localWarning, (in dividebyB) ‘k2‘ is implicitly declared localWarning, (in dividebyB) ‘k3‘ is implicitly declared localWarning, (in dividebyB) ‘t‘ is implicitly declared local

$dividebyB := \boldsymbol{proc}(a, b, c) \quad \boldsymbol{local} \quad oldk1, oldk2, oldk3, k1, k2, k3, t; \quad \boldsymbol{global} \; countB, s; \quad t; \quad oldk1 := a; \quad o$ (11)

$+ 1; \quad k1 := s[1]; \quad k2 := s[2]; \quad k3 := s[3]; \quad s := [divide3(k1, k2, k3, -1, 1, 0)] \quad \mathrm{end\ do}; \quad s; \quad t := s[1], s[2], s[3] \quad \mathrm{end\ proc}$

> $dividebyC := \mathrm{proc}(a, b, c)\, \mathrm{global}\; countC, s, t;\; oldk1 := a{:}\; oldk2 := b{:}\; oldk3 := c{:}\; countC := 0;\; s := [a, b, c];\, ;\, \mathrm{while}\; divisiblebyC(s[1], s[2], s[3])\; \mathrm{do}\; countC := countC + 1;\; k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1, k2, k3, 1, 0, 1)]; \mathrm{od}; s; t := (s[1], s[2], s[3]); \mathrm{end};$

Warning, (in dividebyC) ‘oldk1‘ is implicitly declared localWarning, (in dividebyC) ‘oldk2‘ is implicitly declared localWarning, (in dividebyC) ‘oldk3‘ is implicitly declared localWarning, (in dividebyC) ‘k1‘ is implicitly declared localWarning, (in dividebyC) ‘k2‘ is implicitly declared localWarning, (in dividebyC) ‘k3‘ is implicitly declared local

$dividebyC := \boldsymbol{proc}(a, b, c) \quad \boldsymbol{local} \quad oldk1, oldk2, oldk3, k1, k2, k3; \quad \boldsymbol{global} \; countC, s, t; \quad oldk1 := a; \quad oldk$ (12)

$+1; \quad k1 := s[1]; \quad k2 := s[2]; \quad k3 := s[3]; \quad s := [divide3(k1, k2, k3, 1, 0, 1)] \quad \mathrm{end\ do}; \quad s; \quad t := s[1], s[2], s[$

> $dividebyD := \mathrm{proc}(a, b, c)\, \mathrm{global}\; countD, s, t;\; oldk1 := a{:}\; oldk2 := b{:}\; oldk3 := c{:}\; countD := 0;\; s := [a, b, c];\, ;\, \mathrm{while}\; divisiblebyD(s[1], s[2], s[3])\; \mathrm{do}\; countD := countD + 1;\; k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1, k2, k3, 1, 1, -1)]; \mathrm{od};\; s; t := (s[1], s[2], s[3]); \mathrm{end};$

Warning, (in dividebyD) ‘oldk1‘ is implicitly declared localWarning, (in dividebyD) ‘oldk2‘ is implicitly declared localWarning, (in dividebyD) ‘oldk3‘ is implicitly declared localWarning, (in dividebyD) ‘k1‘ is implicitly declared localWarning, (in dividebyD) ‘k2‘ is implicitly declared localWarning, (in dividebyD) ‘k3‘ is implicitly declared local

$dividebyD := \boldsymbol{proc}(a, b, c) \quad \boldsymbol{local} \quad oldk1, oldk2, oldk3, k1, k2, k3; \quad \boldsymbol{global} \; countD, s, t; \quad oldk1 := a; \quad oldk$ (13)

$+1; \quad k1 := s[1]; \quad k2 := s[2]; \quad k3 := s[3]; \quad s := [divide3(k1, k2, k3, 1, 1, -1)] \quad \mathrm{end\ do}; \quad s; \quad t := s[1], s[2], s[3] \quad \mathrm{end\ proc}$

> $dividebyE := \mathrm{proc}(a,b,c)\,\mathrm{global}\,countE, s, t;\ oldk1 := a\colon oldk2 := b\colon oldk3 := c\colon countE := 0;\ s := [a,b,c];\mathrm{while}\,divisiblebyE(s[1],s[2],s[3])\,\mathrm{do}\,countE := countE + 1;\ k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1,k2,k3,1,-2,0)];\mathrm{od};\ s; t := (s[1],s[2],s[3]);\mathrm{end};$

Warning, (in dividebyE) ‘oldk1‘ is implicitly declared localWarning, (in dividebyE) ‘oldk2‘ is implicitly declared localWarning, (in dividebyE) ‘oldk3‘ is implicitly declared localWarning, (in dividebyE) ‘k1‘ is implicitly declared localWarning, (in dividebyE) ‘k2‘ is implicitly declared localWarning, (in dividebyE) ‘k3‘ is implicitly declared local

$dividebyE := \boldsymbol{proc}(a,b,c)\quad \boldsymbol{local}\quad oldk1, oldk2, oldk3, k1, k2, k3;\quad \boldsymbol{global}\ countE, s, t;\quad oldk1 := a;\quad oldk2$ (14)
$+1;\quad k1 := s[1];\quad k2 := s[2];\quad k3 := s[3];\quad s := [divide3(k1,k2,k3,1,$
$-2,0)]\quad \mathrm{end\ do};\quad s;\quad t := s[1], s[2], s[3]\quad \mathrm{end\ proc}$

> $dividebyF := \mathrm{proc}(a,b,c)\,\mathrm{global}\,countF, s, t;\ oldk1 := a\colon oldk2 := b\colon oldk3 := c\colon countF := 0;\ s := [a,b,c];;\mathrm{while}\,divisiblebyF(s[1],s[2],s[3])\,\mathrm{do}\,countF := countF + 1;\ k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1,k2,k3,3,0,-1)];\mathrm{od};\ s; t := (s[1],s[2],s[3]);\mathrm{end};$

Warning, (in dividebyF) ‘oldk1‘ is implicitly declared localWarning, (in dividebyF) ‘oldk2‘ is implicitly declared localWarning, (in dividebyF) ‘oldk3‘ is implicitly declared localWarning, (in dividebyF) ‘k1‘ is implicitly declared localWarning, (in dividebyF) ‘k2‘ is implicitly declared localWarning, (in dividebyF) ‘k3‘ is implicitly declared local

$dividebyF := \boldsymbol{proc}(a,b,c)\quad \boldsymbol{local}\quad oldk1, oldk2, oldk3, k1, k2, k3;\quad \boldsymbol{global}\ countF, s, t;\quad oldk1 := a;\quad oldk2$ (15)
$+1;\quad k1 := s[1];\quad k2 := s[2];\quad k3 := s[3];\quad s := [divide3(k1,k2,k3,3,0,$
$-1)]\quad \mathrm{end\ do};\quad s;\quad t := s[1], s[2], s[3]\quad \mathrm{end\ proc}$

>

> $dividebyU2 := \mathrm{proc}(a,b,c)\,\mathrm{global}\,countU, s, sign1;\ sign1 := -1;\ \mathrm{if}\ norm1(a,b,c) \neq 1 \wedge norm1(a,b,c) \neq -1\,\mathrm{then}\,false\,\mathrm{else}\,oldk1 := a\colon oldk2 := b\colon oldk3 := c;\ s := [a,b,c];\ \mathrm{if}\ s = [0,0,0]\,\mathrm{then}\,false\,\mathrm{elif}\ s = [-1,1,-1]\,\mathrm{then}\,false\,\mathrm{elif}\ s = [1,-1,1]\,\mathrm{then}\,false\,\mathrm{elif}\ s = [1,0,0]\,\mathrm{then}\,countU := 0; sign1 := 0;\ print(countU); print(sign1);\mathrm{elif}\ s = [-1,0,0]\,\mathrm{then}\,countU := 0; sign1 := 1;\ print(countU); print(sign1);\ \mathrm{else\,if}\ s = [-1,1,-1]\,\mathrm{then}\,false\,\mathrm{else}\,countU := 1;\mathrm{while}\,\neg((s[1] = 1 \wedge s[2] = 1 \wedge s[3] = 0) \vee (s[1] = -1 \wedge s[2] = -1 \wedge s[3] = 0))\,\mathrm{do}\,countU := countU + 1;\ k1 := s[1]; k2 := s[2]; k3 := s[3]; s := [divide3(k1,k2,k3,1,1,0)];\mathrm{od};\mathrm{if}\ k1 = 1\,\mathrm{then}\,sign1 := 1\,\mathrm{else}\,sign1 := 1\,\mathrm{fi};\ \mathrm{fi};\mathrm{fi};\mathrm{fi};\mathrm{end};$

Warning, (in dividebyU2) ‘oldk1‘ is implicitly declared localWarning, (in dividebyU2) ‘oldk2‘ is implicitly declared localWarning, (in dividebyU2) ‘oldk3‘ is implicitly declared localWarning, (in dividebyU2) ‘k1‘ is implicitly declared localWarning, (in dividebyU2) ‘k2‘ is implicitly declared localWarning, (in dividebyU2) 'k3' is implicitly declared local

$dividebyU2 := \textbf{proc}(a,b,c)$ **local** $oldk1, oldk2, oldk3, k1, k2, k3;$ **global** $countU, s, sign1;$ $sign1 :=$ (16)
$-1;$ **if** $norm1(a,b,c) <$
$> 1 \quad \wedge \quad norm1(a,b,c)$
$<$
> -1
then *false* **else** $oldk1 := a;$ $oldk2 := b;$ $oldk3 := c;$ $s := [a,b,c];$ **if** $s = [0,0,0]$ **then** *false*

\>
The factorisation procedure below performs the divisions by each of the factor bases

\> $factorisation := \text{proc}(a,b,c);$ if $a = 0 \wedge b = 0 \wedge c = 0$ then *false* else $dividebyU2(dividebyA(dividebyB(divideby C$
0 then $print(a,b,[sign1, countU, countA, countB, countC, countD, countE, countF])$ else $print(a,b, False\ -\ does$

$factorisation :=$ (17)
$\textbf{proc}(a,b,c)$
if $a = 0 \quad \wedge \quad b$
$= 0 \quad \wedge \quad c$
$= 0$ **then** *false* **else**
$dividebyU2(dividebyA(dividebyB(dividebyC(dividebyD(dividebyE(dividebyF(a,b,c)))))))$
;
if 0
$<$
$= sign1$ **then**
$print(a,b,$
$[sign1, countU, countA, countB, countC, countD, countE,$
$countF])$
else $print(a,b, False\ -\ does\ not\ factor);$ *false* end if
end if
end proc

\>
The next two procedures perform multiplication of triples [a,b,c]; mult1 combines a pair of tripes while mult4 takes a string and uses mult1 to combines them pairwise
\>
\> $mult1 := \text{proc}(x,y);\ mult2(x[1], x[2], x[3], y[1], y[2], y[3]);$ end;

$$mult1 := \textbf{proc}(x,y)\quad mult2(x[1], x[2], x[3], y[1], y[2], y[3]) \text{ end proc} \tag{18}$$

>
>
> $mult4 := \mathrm{proc}()\mathrm{global}\, L;\ L := [];\ \mathrm{for}\, i\, \mathrm{from}\, 1\, \mathrm{to}\, nargs\, \mathrm{do}\, L := [op(L), args[i]]; \mathrm{od};\ \mathrm{if}\ nops(L) = 2\, \mathrm{then}\, mult1(op(1, L), op(2, L))\, \mathrm{else}\, k := [mult1(op(1, L), op(2, L))]; L := subsop(1 = NULL, L);\ L := subsop(1 = NULL, L); L := [k, op(L)];\ mult4(op(L)); \mathrm{fi}; \mathrm{end};$
Warning, (in mult4) ‘i‘ is implicitly declared localWarning, (in mult4) ‘k‘ is implicitly declared local

$$mult4 := \tag{19}$$
$\boldsymbol{proc}()\quad \boldsymbol{local}\quad i, k;\quad \boldsymbol{global}\quad L;$
$L := [\,];\quad \boldsymbol{for}\quad i\quad \boldsymbol{to}\quad nargs\quad \boldsymbol{do}\quad L := [op(L), args[i]]\quad \text{end do};$
$\boldsymbol{if}\quad nops(L) = 2\quad \boldsymbol{then}\quad mult1(op(1, L), op(2, L))\quad \boldsymbol{else}$
$k := [mult1(op(1, L), op(2, L))];\quad L := subsop(1 = NULL, L);\quad L := subsop(1 = NULL, L);\quad L := [k, op(1$

> $mult4(U, U, B, E);$

$$1, 5, 3 \tag{20}$$

>
>
Now we use these procedures to try to factorise N = 9263 = 59*157 = 21ˆ3 + 2. Hence r = 21 and we can work in the field Q $((-2)^{\frac{1}{3}})$whoseprimeswehavestudied
> $N := 21^3 + 2;$

$$N := 9263 \tag{21}$$

> $ifactor(N);$

$$(59)\,(157) \tag{22}$$

In the next step we populate a list B with the values of a and b (small) where the factor base for a + 21b contains only the small primes {2,3,5,7,11,13}
There are 47 pairs (a,b) with a and b between -9 and 9 where both a + 21b and [a,b,0] can be completely factorised using a small factor base. The values of the factors form a 15 column matrix; the first 7 columns are the powers of -1, 2, 3, 5, 7, 11, 13 that factor a + 21b and the last 8 are the powers of -1, U, A, B, C, D, E, F that factorise [a,b,0]. To display the matrix we need to increase the default size to 50x50.
> $interface(rtablesize = 50)$

$$50 \tag{23}$$

>
Row a b a+21b
1 -9 -3 -72
2 -9 0 -9
3 -7 -3 -70

4 -7 1 14
5 -6 -4 -90
6 -6 -2 -48
7 -6 0 -6
8 -4 -4 -88
9 -4 4 80
10 -3 -3 -66
11 -3 -2 -45
12 -3 -1 -24
13 -3 3 60
14 -2 -3 -65
15 -2 -2 -44
16 -2 0 -2
17 -2 2 40
18 -1 -1 -22
19 -1 0 -1
20 0 -4 -84
21 0 -3 -63
22 0 -1 -21
23 0 2 42
24 0 3 63
25 9 9 198
26 0 4 84
27 1 -1 -20
28 1 1 22
29 2 -2 -40
30 2 2 44
31 2 3 65
32 3 -3 -60
33 3 1 24
34 3 3 66
35 4 -4 -80
36 4 4 88
37 6 0 6
38 6 2 48
39 6 4 90
40 7 -1 -14
41 7 3 70
42 9 0 9
43 9 3 72
44 -9 9 180
45 -8 8 160
46 6 6 132
47 8 8 176

> $R := Matrix(47, 15, [1, 3, 2, 0, 0, 0, 0, 1, 2, 0, 3, 2, 0, 0, 0, 1, 0, 2, 0, 0, 0, 0, 1, 2, 0, 6, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0$

$$
R \coloneqq
\begin{bmatrix}
1 & 3 & 2 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 3 & 2 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 6 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 1 & 2 & 1 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & 0 & 1 & 0 & 0 \\
1 & 4 & 3 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & 2 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 3 & 0 & 0 & 0 & 0 \\
1 & 3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 6 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 3 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 2 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 4 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 2 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 2 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 3 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 3 & 0 & 1 & 0 & 0 \\
0 & 2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 2 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 3 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 3 & 0 & 0 & 0 & 0 \\
1 & 4 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 6 & 1 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 3 & 0 & 0 & 0 & 0 \\
0 & 4 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & 2 & 0 & 0 & 0 \\
0 & 1 & 2 & 1 & 0 & 0 & 0 & 1 & 1 & 3 & 0 & 0 & 1 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 6 & 0 & 0 & 0 & 0 \\
0 & 3 & 2 & 0 & 0 & 0 & 0 & 1 & 2 & 0 & 3 & 2 & 0 & 0 & 0 \\
0 & 2 & 2 & 1 & 0 & 0 & 0 & 1 & 2 & 0 & 7 & 0 & 0 & 0 & 0 \\
0 & 5 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 9 & 1 & 0 & 0 & 0 & 0 \\
0 & 2 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 3 & 3 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 9 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\tag{24}
$$

We aim to find sets of rows that are linearly dependent modulo 2; one possibility is rows 23, 37, 41, 45

These rows give us a factor (59) of N when we calculate the values of u and v that give a congruence uˆ2 = vˆ2 mod N and find the gcd of N and u - v