Unless otherwise stated F denotes an arbitrary field.

Algorithm: `UnivariateDivision` (Version I: Long division)

Input: $f, g \in F[x]$ with $g \neq 0$.
Output: $q, r \in F[x]$ such that $f = qg + r$ with $r = 0$ or $\deg r < \deg g$.
BEGIN
$q := 0;$
$r := f;$
while $\deg r \not< \deg g$ do
    (* Recall that $\deg 0 < \deg p$ for all non-zero polynomials $p \in F[x]$. *)
    (* Loop invariant: $f = qg + r$. *)
    $t := \operatorname{lt} r / \operatorname{lt} g;$ (* $\operatorname{lt}(tg) = \operatorname{lt} r$ *)
    $q := q + t;$
    $r := r - tg;$
end while;
return q, r;
END

Algorithm: `UnivariateDivision` (Version II)

Input: $f, g \in F[x]$ with $g \neq 0$.
Output: $q, r \in F[x]$ such that $f = qg + r$ with $r = 0$ or $\deg r < \deg g$.

```
BEGIN
q := 0;
r := 0;
p := f;
while p ≠ 0 do
    (* Loop invariant: f = p + qg + r. *)
    if lt g | lt p then          (* Equivalent condition: deg g ⩽ deg p. *)
        t := lt p/ lt g;         (* lt(tg) = lt p. *)
        q := q + t;
        p := p − tg;
    else
        r := r + lt p;
        p := p − lt p;
    end if;
end while;
return q, r;
END
```

Algorithm: `EuclideanAlgorithm` (Extended; Version I)

Input: $f, g \in F[x]$.
Output: $h, a, b \in F[x]$ such that $h = af + bg$ and h is a g.c.d. of f and g).
BEGIN
$p_0 := f$; $p_1 := g$;
$a_0 := 1$; $a_1 := 0$;
$b_0 := 0$; $b_1 := 1$;
$j := 1$;
while $p_j \neq 0$ do
    (∗ Invariant: $p_i = a_i f + b_i g$ for all i. ∗)
    $q_j, p_{j+1} :=$ `UnivariateDivision`(p_{j-1}, p_j);
    (∗ Quotient q_j, remainder p_{j+1}. We have $p_{j+1} = p_{j-1} - q_j p_j$. ∗)
    $a_{j+1} := a_{j-1} - q_j a_j$;
    $b_{j+1} := b_{j-1} - q_j b_j$;
    $j := j + 1$;   (∗ Writable as $j+ := 1$; ∗)
end while;
$h := p_{j-1}$; $a := a_{j-1}$; $b := b_{j-1}$;
return h, a, b;
END

Algorithm: `EuclideanAlgorithm` (Extended; Version II)

Input: $f, g \in F[x]$.
Output: $h, a, b \in F[x]$ such that $h = af + bg$ and h is a g.c.d. of f and g).
BEGIN
$p_0 := f$; $p_1 := g$;
$a_0 := 1$; $a_1 := 0$;
$b_0 := 0$; $b_1 := 1$;
while $p_1 \neq 0$ do
    (∗ Invariant: $p_i = a_i f + b_i g$ for all i. ∗)
    $q, p_2 :=$ `UnivariateDivision`(p_0, p_1);
    (∗ Quotient q, remainder p_2. We have $p_2 = p_0 - qp_1$. ∗)
    $a_2 := a_0 - qa_1$;
    $b_2 := b_0 - qb_1$;
    (∗ Now to place the current working values for the ps, namely
        p_1 and p_2, into p_0 and p_1. Similarly for the as and bs. ∗)
    $p_0 := p_1$; $a_0 := a_1$; $b_0 := b_1$;
    $p_1 := p_2$; $a_1 := a_2$; $b_1 := b_2$;
    (∗ Above two lines *cannot* be swapped. ∗)
end while;
$h := p_0$; $a := a_0$; $b := b_0$;
return h, a, b;
END

Algorithm: `MultivariateDivision`

Input: $f, f_1, f_2, \ldots, f_s \in F[x_1, x_2, \ldots, x_n]$ such that $0 \notin \{f_1, f_2, \ldots, f_s\}$.
Output: $q_1, q_2, \ldots, q_s, r \in F[x_1, x_2, \ldots, x_n]$ such that

(i) $f = q_1 f_1 + \cdots + q_s f_s + r$, and
for $1 \leqslant i \leqslant s$ if $q_i f_i \neq 0$ then $\operatorname{mdeg} f \geqslant \operatorname{mdeg}(q_i f_i)$

(ii) $r = 0$ or $r = \sum_{\beta \in B} b_\beta \underline{x}^\beta$, where $0 \neq b_\beta \in F$ for all $\beta \in B$
and $\operatorname{lt} f_i \nmid b_\beta \underline{x}^\beta$ for each $\beta \in B$ and $i \in \{1, 2, \ldots, s\}$.

BEGIN
for $i \in \{1, 2, \ldots, s\}$ do
    $q_i := 0$;
end for;
$r := 0$;
$p := f$;
while $p \neq 0$ do
    (∗ Loop invariant: $f = p + q_1 f_1 + \cdots + q_s f_s + r$. ∗)
    if there exists $j \in \{1, 2, \ldots, s\}$ such that $\operatorname{lt} f_j \mid \operatorname{lt} p$ then
        $i :=$ least $j \in \{1, 2, \ldots, s\}$ such that $\operatorname{lt} f_j \mid \operatorname{lt} p$;
        $t := \operatorname{lt} p / \operatorname{lt} f_i$;   (∗ $\operatorname{lt}(t f_i) = t \operatorname{lt} f_i = \operatorname{lt} p$. ∗)
        $q_i := q_i + t$;
        $p := p - t f_i$;
    else
        $r := r + \operatorname{lt} p$;
        $p := p - \operatorname{lt} p$;
    end if;
end while;
return $q_1, q_2, \ldots, q_s, r$;
END

Algorithm: `GröbnerBasis`

Input: Non-zero $f_1, f_2, \ldots, f_s \in R = F[x_1, x_2, \ldots, x_n]$
and a monomial order $\leqslant$ for R.
Output: A Gröbner basis G for $\langle f_1, f_2, \ldots, f_s \rangle$
with respect to $\leqslant$ such that $f_1, f_2, \ldots, f_s \in G$.

```
BEGIN
G := {f_1, f_2, ..., f_s};
repeat
    T := {}; t := |G|;
    order the elements of G somehow as g_1, ..., g_t;
    for i := 1, ..., t − 1 do
        for j := i + 1, ..., t do
            r := S(g_i, g_j) rem (g_1, ..., g_t);
            if r ≠ 0 then
                T := T ∪ {r};
            end if;
        end for;
    end for;
    G := G ∪ T;
until T = {};
return G;
END
```

Algorithm: `ReducedGröbnerBasis`

Input: A Gröbner basis $H = \{h_1, \dots, h_s\}$ w.r.t. $\leqslant$ for the ideal
I of $R = F[x_1, x_2, \dots, x_n]$ and the monomial order $\leqslant$ of R.
Output: The reduced Gröbner basis $G = \{g_1, \dots, g_m\}$ for I w.r.t. $\leqslant$.

```
BEGIN
G := {h_1, h_2, ..., h_s};
order G = {g_1, ..., g_s} so that lt g_1 ⩽ lt g_2 ⩽ ··· ⩽ lt g_s.
for j := s, s − 1, ..., 1 do
    if exists k ∈ { i : 1 ⩽ i ⩽ j − 1 | (lt g_i | lt g_j) } then
        remove g_j from G;
    end if;
end for;
rename the g_i so that G = {g_1, ..., g_m} (m ⩽ s) and lt g_1 ⩽ ··· ⩽ lt g_m;
for j := 1, ..., m do g_j := g_j/(lc g_j); end for;
for j := 2, ..., m do
    flag := true;
    while flag do
        if exists term t of g_j − lt g_j such that lt g_i | t for some i then
            choose such t so that lm t is maximal w.r.t. ⩽;
            let i be minimal such that lt g_i | t;
            (* We have 1 ⩽ i < j. Also t = lt t. *)
            g_j := g_j − (t/ lt g_i)g_i;
        else
            flag := false;
        end if;
    end while;
end for;
return G = {g_1, ..., g_m};
END
```

Algorithm: `GramSchmidtOrthogonalisation`

Input: Linearly independent vectors $v_1, \ldots, v_r \in F^n$ where $r \leqslant n$, $F \subseteq \mathbb{C}$.
Output: Vectors $v_1^*, \ldots, v_r^* \in F^n$ satisfying $v_i^* \cdot v_j^* = 0$ for $1 \leqslant i < j \leqslant r$
such that $v_k - v_k^* \in \langle v_1, \ldots, v_{k-1} \rangle_F$ for $1 \leqslant k \leqslant r$.

BEGIN
$v_1^* := v_1$; (* If $r \neq 0$. *)
for $i := 2, \ldots, r$ do
  for $j := 1, \ldots, i-1$ do
    $\mu_{ij} := (v_i \cdot v_j^*)/(v_j^* \cdot v_j^*)$;
  end for;
  (* We can also define $\mu_{ii} = 1$ and $\mu_{ij} = 0$ if $j > i$. *)
  $v_i^* := v_i - \sum_{j=1}^{i-1} \mu_{ij} v_j^*$;
end for;
return $v_1^*, v_2^*, \ldots, v_r^*$;
END

Algorithm: `BasisReduction` (Variant of LLL)

Input: $\mathbb{R}$-linearly independent vectors $v_1, \ldots, v_n \in \mathbb{Z}^n$.
Output: A reduced basis $(w_1, \ldots, w_n)$ of the lattice $L = \langle v_1, \ldots, v_n \rangle_{\mathbb{Z}}$ such that if $((w_1^*, \ldots, w_n^*), (\mu_{st}))$ is the Gram–Schmidt orthogonalisation of $(w_1, \ldots, w_n)$ then $|\mu_{st}| \leqslant \frac{1}{2}$ for $1 \leqslant t < s \leqslant n$.

BEGIN
$(w_1, \ldots, w_n) := (v_1, \ldots, v_n)$;
$((w_1^*, \ldots, w_n^*), (\mu_{st})) :=$ `GramSchmidtOrthogonalisation`$(w_1, \ldots, w_n)$;
$i := 2$;
while $i \leqslant n$ do
    for $j := i-1, i-2, \ldots, 1$ do
    (∗ Loop invariant: $(w_1, \ldots, w_n)$ is a basis for L and has GSO
            $((w_1^*, \ldots, w_n^*), (\mu_{st}))$, where $|\mu_{il}| \leqslant \frac{1}{2}$ for $l = j+1, \ldots, i-1$. ∗)
        $m :=$ Round(μ_{ij});   (∗ so $m \in \mathbb{Z}$ and $|\mu_{ij} - m| \leqslant \frac{1}{2}$. ∗)
        $w_i := w_i - m w_j$;
        (∗ Does not change $\langle w_1, \ldots, w_m \rangle_{\mathbb{Z}}$ or any of the w_k^* in the GSO. ∗)
        for $l := 1, \ldots, j$ do
            $\mu_{il} := \mu_{il} - m\mu_{jl}$;
        end for;
    end for;
    (∗ Now $|\mu_{il}| \leqslant \frac{1}{2}$ for $l = 1, 2, \ldots, i-1$. ∗)
    if $i > 1$ and $|w_{i-1}^*|^2 > 2|w_i^*|^2$ then
        Exchange the values of w_{i-1} and w_i;
        Update $((w_1^*, \ldots, w_n^*), (\mu_{st}))$ to be the GSO of $(w_1, \ldots, w_n)$;
        (∗ Think about how to do this efficiently:
            only w_{i-1}^*, w_i^* and various of the μ_{jk} get altered. ∗)
        $i := i - 1$;
    else
        $i := i + 1$;
    end if;
end while;
return $(w_1, \ldots, w_n)$;
END