In [7]:
```
import numpy as np
from copy import deepcopy
from math import ceil, floor

class Bqf:
    def __init__(self, a, b, c):
        self.terms = [["x^2", a, lambda x, y: x^2],
                      ["xy", b, lambda x, y: x * y],
                      ["y^2", c, lambda x, y: y]]

        self.update_vals(a,b,c)

        self.mat = [
            [self.a, self.b / 2],
            [self.b / 2, c]
        ]

        self.det  = self.determinant()
        self.disc = self.discriminant()

        if (self.disc == 0): self.type = "Perfect square"

        elif (self.disc < 0):
            if (self.a > 0): self.type = "Positive definite"
            else:            self.type = "Negative definite"

        else:                self.type = "Indefinite"

        self.transform = []

    def __call__(self, x, y):
        return sum([i[1] * i[2](x,y) for i in self.terms])

    def __repr__(self):
        out = ""
        for i in range(len(self.terms)):
            if self.terms[i][1] != 0:

                if i > 0:
                    if self.terms[i][1] > 0:
                        out += "+ "
                    else:
                        out += "- "
                elif (self.terms[i][1] < 0):
                    out += "-"

                if abs(self.terms[i][1]) != 1:
                    out += str(abs(self.terms[i][1]))

                out += self.terms[i][0] + " "

        return out[:-1]

    def __eq__(self, other):
        return np.allclose(self.reduce().mat, other.reduce().mat)

    def determinant(self):
        return (self.mat[0][0] * self.mat[1][1]) - (self.mat[0][1] * self.mat
```

```
[1][0])

    def discriminant(self):
        return self.determinant() * -4

    def is_reduced(self):
        return -abs(self.a) < self.b <= abs(self.a) < abs(self.c) or\
                0 < self.b <= abs(self.a) == abs(self.c)

    def update_vals(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

        self.terms[0][1] = a
        self.terms[1][1] = b
        self.terms[2][1] = c

    def reduce(self):
        if (self.is_reduced()):
            print("already reduced")
            return self

        red = deepcopy(self)

        while (not red.is_reduced()):
            print(red.mat)

            if (abs(red.a) > abs(red.c)):
                red.mat = np.matmul(red.mat, [[0, 1],[-1, 0]])
                print([[0, -1],[1, 0]])
            else:
                m = 0
                # find transformation matrix
                if (red.a > 0):
                    m = floor((abs(red.a) - red.b)/(2*red.a))
                else:
                    m = ceil((abs(red.a) - red.b)/(2*red.a))

                red.mat = np.matmul(red.mat, [[1, m],[0, 1]])
                print([[1, m],[0, 1]])

            #assert red.mat[0][1] * 2 == red.b

            if (abs(red.a) == abs(red.c) and red.b < 0):
                red.mat = np.matmul(red.mat, [[0, 1],[-1,0]])

                red.update_vals(red.mat[0][0], red.mat[0][1] * 2, red.mat
[1][1])

        print(red.mat)
        print("is reduced")
        assert red.mat[0][1] * 2 == red.b
        print(red.mat[0][1])
        print(red)
        return red
```

3.4

1. For each of the following, determine whether the form is positive definite, negative definite, or indefinite.

(a) $x^2 + y^2$

(b) $-(x^2) - (y^2)$

(c) $x^2 - (2y^2)$

(d) $10x^2 - (9xy) + (8y^2)$

(e) $x^2 - (3xy) + y^2$

(f) $17x^2 - (26xy) + 10y^2$

In [8]:
```
bqfs_1 = [
    (1, 0, 1),
    (-1, 0, -1),
    (1, 0, -2),
    (10, -9, 8),
    (1, -3, 1),
    (17, -26, 10)
]

for i in range(len(bqfs_1)):
    bqf = Bqf(*bqfs_1[i])
    print("({}) {} is {}".format(chr(ord('`')+ i + 1), bqf, bqf.type))
```

```
(a) x^2 + y^2 is Positive definite
(b) -x^2 - y^2 is Negative definite
(c) x^2 - 2y^2 is Indefinite
(d) 10x^2 - 9xy + 8y^2 is Positive definite
(e) x^2 - 3xy + y^2 is Indefinite
(f) 17x^2 - 26xy + 10y^2 is Positive definite
```

2. Prove that the quadratic form $x^2 - (2xy) + y^2$ has discriminant 0. Determine the class of integers represented by this form.

In [9]:
```
bqf = Bqf(1, -2, 1)

assert bqf.det == 0
print("{}'s determinant is {}".format(bqf, int(bqf.det)))
```

```
x^2 - 2xy + y^2's determinant is 0
```

$x^2 - (2xy) + y^2 = (x - y)^2$. Because the domain of x and y are all integers, the BQF can represent all perfect squares.

4. Use the binomial theorem to give a formula for positive integers x_k and y_k such that $(3 + 2\sqrt{2})^k = x_k + y_k\sqrt{2}$.

$x_k = \Sigma_{n=0}^{\lfloor k/2 \rfloor} \binom{k}{n} 3^{k-2n} 2^{3n}$

$y_k = \Sigma_{n=0}^{\lfloor k/2 \rfloor} \binom{k}{n} 3^{k-(2n+1)} 2^{3n+1}$

(See attached document for work)

Show that $(3 - 2\sqrt{2})^k = x_k - y_k\sqrt{2}$.

Because x_k represents all even powers in the binomial expansion of the expression, any negations will not effect its value, maintaining its positive status as x_k.

y_k, however, represents the odd powers, so its terms will be negated as shown in the above expression $(-y_k)$.

Deduce that $x_k^2 - (2y_k^2) = 1$ for $k = 1, 2, 3, \cdots$.

Let us prove this by induction.

$x_1^2 - (2y_1^2) = 9 - 8 = 1$

Let's assume that this works for x_n and y_n. Now, we will prove it for x_{n+1} and y_{n+1}.

As will be proved later, $x_{n+1} = 3x_n + 4y_n$ and $y_{n+1} = 2x_n + 3y_n$

We can write $x_{n+1}^2 - (2y_{n+1}^2)$ as $9x_n^2 + 24x_n y_n + 16y_n^2 - (2(4x_n^2 + 12x_n y_n + 9y_n^2)) = x_n^n - 2y_n^2$ which equals 1 by the inductive assumption.

Therefore, $x_k^2 - (2y_k^2) = 1$ by induction.

Show that $(x_k, y_k) = 1$ for each k.

Let us assume that $(x_k, y_k) \neq 1$

It is known that $x_k^2 = (2y_k^2) + 1$.

$\implies x_k \cdot x_k = y_k \cdot y_k + 1$

$\implies x_k = \frac{y_k \cdot y_k}{x_k} + \frac{1}{x_k}$

Note that $x_k \mid y_k$, but $x_k \nmid 1$ (unless x_k is 1, in which the GCD is trivially 1).

Therefore, $\frac{y_k \cdot y_k}{x_k} + \frac{1}{x_k} \notin \mathbb{Z}$.

Our assumption must be false, as $x_k \in \mathbb{Z}$.

$(x_k, y_k) = 1$.

Show that $x_{k+1} = 3x_k + 4y_k$ and $y_{k+1} = 2x_k + 3y_k$ for $k = 1, 2, 3, \cdots$.

We can find the next iteration of this binomial by observation:

First, it is important to acknowledge that $(3 + 2\sqrt{2})^{k+1} = (3 + 2\sqrt{2})^k(3 + 2\sqrt{2})$,

So we can just examine what would happen when multiplying $(3 + 2\sqrt{2})^k$ by $(3 + 2\sqrt{2})$.

For the integer term of the expanded binomial, we must consider the integer term of the multiplied binomial, 3, as multiplied by the "old" integer term, and the two rational terms multiplied together.

That is, $x_{k+1} = x_k \cdot 3 + y_k\sqrt{2} \cdot 2\sqrt{2} = 3x_k + 4y_k$.

Similarly, y_{k+1} is composed of the "old" integer term multiplied with the multiplied binomial's rational term added with the "old" rational term multiplied with the multiplied binomial's integer term.

So, $y_{k+1} = x_k \cdot 2 + y_k \cdot 3$.

Show that $\{x_k\}$ and $\{y_k\}$ are strictly increasing sequences.

Because $x_{k+1} = 3x_k + 4y_k$, $y_{k+1} = 2x_k + 3y_k$, and $x_k, y_k \geq 0$, their sequence representations must be strictly increasing.

Conclude that the number 1 has infinitely many proper representations by the quadratic form $x^2 - (2y^2)$.

Because we've proved that $x_k^2 - (2y_k^2) = 1$ for $k = 1, 2, 3, \cdots$, there are infinitely many k, and $(x_k, y_k) = 1$, we can conclude that there are infinitely many proper representations of 1 in the BQF.

3.5

1. Find a reduced form that is equivalent to the form $7x^2 + 25xy + 23y^2$.

If we substitute $x = 2x - y$ and $y = -x + y$, we get the form $x^2 + xy + 5y^2$.

Note that this transformation is unimodular, so both forms represent the same class of integers.

Furthermore, notice that the resultant form is reduced.

5. Show that $x^2 + 5y^2$ and $2x^2 + 2xy + 3y^2$ are the only reduced quadratic forms of discriminant -20.

Find all reduced BQFs where $d = b^2 - 4ac = -20$

Case 1: $-|a| < b <= |a| < c$

$b^2 = -20 + 4ac$

$\implies -(a^2) < -20 + 4ac <= (a^2) < c^2$

Only integer solutions are $a = 1, b = 0, c = 5$ and $a = 2, b = 2, c = 3$ (found using computer algebra system)

Case 2: $0 < b \leq a = |c|$

$b^2 = -20 + 4ac$

$\implies 0 < -20 + 4a^2 \leq a^2$

$\implies \sqrt{5} < a \leq 2\sqrt{\frac{5}{3}}$

There are no integer solutions of a, so Case 2 yields no BQFs.

Show that the first of these forms does not represent 2, but that the second one does.

First, we will consider $x^2 + 5y^2 = 2$.

Case 1: $x = 0$

$5y^2 = 2$ clearly has no integer solutions, so this case cannot provide a representation.

Case 2: $x = 1$

$5y^2 + 1 = 2$ clearly has no integer solutions, so this case cannot provide a representation.

Case 3: $x > 1$

$\implies x^2 > 2$, and $5y^2 > 0$, so $x^2 + 5y^2 > 2$, meaning this case cannot provide a representation.

==========

Now, we will consider $2x^2 + 2xy + 3y^2$.

$x = 1, y = 0$ yields $2(1)^2 + 2(1)(0) + (0)^2 = 2$.

Deduce that these forms are inequivalent, and hence that $H(-20) = 2$.

Because $A = x^2 + 5y^2 = 2$ does not represent 2, but $B = 2x^2 + 2xy + 3y^2$ does, they cannot be equivalent - there is no matrix M such that $A = M^T B M$

Show that an odd prime p is represented by at least one of these forms if and only if $p \equiv 1, 3, 7,$ or $9 \pmod{20}$.

BQFs A and B are the only reduced BQFs with discriminant -20 (as shown previously). Because $-20 = 0 \mod 4$, we can use Corollary 3.14 to say that p is represented by either A or $B \iff (\frac{-20}{p}) = 1$.

In [11]:
```
# Short prime finder
prime_list = [2] + [*filter(lambda i:all(i%j for j in range(3,i,2)), range(3,10
000,2))]

possible = set()
not_possible = set()

def legendre(a, p):
    return ((a) ** ((p - 1) // 2)) % p

for j in range(20):
    for i in prime_list:
        if (i % 20 == j and i % 2 != 0):
            if (legendre(-20, i) == 1):
                possible.add(j)
            else:
                not_possible.add(j)
            break;

print("Odd primes equivalent to the following mod 20 are acheivable from a BQF
with discriminant -20:")
print(str(possible)[1:-1])
print("Odd primes equivalent to the following mod 20 are not acheivable from a
BQF with discriminant -20:")
print(str(not_possible)[1:-1])
print("There are no odd primes equivalent to the following mod 20:")
print(str(set([i for i in range(20)]) - (possible | not_possible))[1:-1])
```

```
Odd primes equivalent to the following mod 20 are acheivable from a BQF with d
iscriminant -20:
1, 3, 9, 7
Odd primes equivalent to the following mod 20 are not acheivable from a BQF wi
th discriminant -20:
5, 11, 13, 17, 19
There are no odd primes equivalent to the following mod 20:
0, 2, 4, 6, 8, 10, 12, 14, 15, 16, 18
```