In [1]:
```python
def gauss_seidel(A, b):
    solution = [0, 0]
    for _ in range(100):
        for k in range(2):
            sum_of_other_side = b[k]
            for j in range(2):
                if k != j:
                    sum_of_other_side -= solution[j] * A[k][j]
            solution[k] = sum_of_other_side / A[k][k] # dividing by the coefficient
    return solution
```

Part A

Giving x-2y=4,2x+y=3 to Wolfram leads to:

$x \approx 2, y \approx -1$

And with our code:

In [2]:
```python
gauss_seidel([[1, -2], [2, 1]], [4, 3])
```

Out[2]:
```
[-8.0346902212949505e+59, 1.6069380442589901e+60]
```

We will describe the reason of the wrong answer in part C.

Part B

Giving 2x+y=3,x-2y=4 to Wolfram leads to:

$x \approx 2, y \approx -1$

And with our code:

In [3]:
```python
gauss_seidel([[2, 1], [1, -2]], [3, 4])
```

Out[3]:
```
[2.0, -1.0]
```

So we get the same results: x = 2 and y = -1.

Part C

The results from part A are wrong, because the matrix is not diagonally dominant (for example, $1 < |2|$ in the first row). So the Gauss Seidel method can diverge (and in this case, it doesn't converge, based on what we found from running the code above). But for part B it is diagonally dominant, so the method converges.

The matrix from part A:

$$\begin{bmatrix} 1 & -2 \\ 2 & 1 \end{bmatrix}$$

The matrix from part B:

$$\begin{bmatrix} 2 & 1 \\ 1 & -2 \end{bmatrix}$$