Python program implementing the extended binary GCD algorithm.

```
def ext_binary_gcd(a,b):
    """Extended binary GCD.

    Given input a, b the function returns d, s, t
    such that gcd(a,b) = d = as + bt."""
    u, v, s, t, r = 1, 0, 0, 1, 0
    while (a % 2 == 0) and (b % 2 == 0):
        a, b, r = a//2, b//2, r+1
    alpha, beta = a, b
#
# from here on we maintain a = u * alpha + v * beta
# and b = s * alpha + t * beta
#
    while (a % 2 == 0):
        a = a//2
        if (u % 2 == 0) and (v % 2 == 0):
            u, v = u//2, v//2
        else:
            u, v = (u + beta)//2, (v - alpha)//2
    while a != b:
        if (b % 2 == 0):
            b = b//2
#
# Commentary: note that here, since b is even,
# (i) if s, t are both odd then so are alpha, beta
# (ii) if s is odd and t even then alpha must be even, so beta is odd
# (iii) if t is odd and s even then beta must be even, so alpha is odd
# so for each of (i), (ii) and (iii) s + beta and t - alpha are even
#
            if (s % 2 == 0) and (t % 2 == 0):
                s, t = s//2, t//2
            else:
                s, t = (s + beta)//2, (t - alpha)//2
        elif b < a:
            a, b, u, v, s, t = b, a, s, t, u, v
        else:
            b, s, t = b - a, s - u, t - v
    return (2 ** r) * a, s, t
```

Alternative Python program implementing the extended binary GCD algorithm.

```
#!/usr/local/bin/Python

def strip_powers_of_two(c, p, q, gamma, delta):
    c = c / 2
    if (p % 2 == 0) and (q % 2 == 0):
        p, q = p//2, q//2
    else:
        p, q = (p + delta)//2, (q - gamma)//2
    return c, p, q

def ext_bin_gcd(a,b):
    """Extended binary GCD.

    Given input a, b the function returns d, s, t
    such that gcd(a,b) = d = as + bt."""
    u, v, s, t, r = 1, 0, 0, 1, 0
    while (a % 2 == 0) and (b % 2 == 0):
        a, b, r = a//2, b//2, r+1
    alpha, beta = a, b
   while (a % 2 == 0):
        a, u, v = strip_powers_of_two(a, u, v, alpha, beta)
    while a != b:
        if (b % 2 == 0):
            b, s, t = strip_powers_of_two(b, s, t, alpha, beta)
        elif b < a:
            a, b, u, v, s, t = b, a, s, t, u, v
        else:
            b, s, t = b - a, s - u, t - v
    return (2 ** r) * a, s, t
```

Python programs implementing Euclid's algorithm for computing the GCD and extended GCD.

```
#!/usr/local/bin/Python
def euclid(a,b):
    """Euclid's algorithm for GCD.

    Given input a, b the function returns d such that gcd(a,b) = d."""
    if a < b:
        a, b = b, a
    else:
        pass
    while b != 0:
        a, b = b, a % b
    return a

def ext_euclid(a,b):
    """Extended Euclid's algorithm for GCD.

    Given input a, b the function returns d such that gcd(a,b) = d
    and x, y such that ax + by = d, as well as u, v such that au = bv."""
    if a < b:
        a, b = b, a
    else:
        pass
    u, v, x, y = 0, 1, 1, 0
    while b != 0:
        a, b, x, y, u, v = b, a % b, u, v, x - ( a // b ) * u, y - ( a // b ) * v
    return a, x, y, u, v
```