Answer

Andy and Candy are both correct but for the post condition to hold it is required that a and b be greater than or equal to 0 as the rules are defined as such.

```
GCD1 terminates with
    r == gcd(a,b) as the post condition
    a > 0 && b > 0 as the pre condition
    decreases b as the termination metric

GCD2 terminates with
    r == gcd(a,b) as the post condition
    a >= 0 && b >= 0 as the pre condition
    decreases b as the termination metric
```

GCD1

```
method GCD1(a: int, b: int) returns (r: int)
requires a > 0 && b > 0
decreases b
//ensures r == gcd(a,b)
{

/*
    a > 0 && b > 0
*/

/* BRING ALL CASES TOGETHER
    (a < b && a > 0 && b > 0)
    ||
    (a > 0 && b > 0 && a >= b && a % b == 0)
    ||
    (a > 0 && b > 0 && a >= b)
*/

// CASE 3
/* // + CASE a < b +

        (a < b && a > 0 && b > 0)

    <===>
        ( a < b ) && (
            ( b > 0 && a > 0 ) &&
            ( TRUE )
        )

    <===>
        ( a < b && ( (a % b) != 0 ) ) && (
            ( b > 0 && a > 0 ) &&
            ( gcd(b,a) == gcd(b,a) )
        )

    <===> + REPEATED so REMOVED +
        ( a < b ) && (
            ( b > 0 && a > 0 ) &&
            ( gcd(b,a) == gcd(a,b) )
        )
        &&
        ( a < b ) && (
            ( b > 0 && a > 0 ) &&
            ( gcd(b,a) == gcd(a,b) )
        )
*/ // + CASE a < b +
// CASE 3
```

```
// CASE 2
/* // + CASE a >= b && a % b == 0 +

    a > 0 && b > 0 && a >= b && a % b == 0

    <===> +  a >= 0 && b > 0 && a >= b SO a > 0 +
    a >= 0 && b > 0 && a >= b && a % b == 0
            && TRUE

    <===> + SIMPLIFY +
    a >= 0 && b > 0 && a >= b && a % b == 0
            && b == b

    <===> + APPLY RULE v. (defined below) +
        a >= 0 && b > 0 && a >= b && a % b == 0
            && b == gcd(a,b)

    <===> + ASSUME b > 0 && a >= 0 to apply rules +
        a >= b && a % b == 0
            && b == gcd(a,b)

*/ // + CASE a >= b && a % b == 0 +
// CASE 2

// CASE 1
/* // CASE a >= b && a % b != 0

    (a > 0 && b > 0 && a >= b)

    <===>
         (a > 0 && b > 0 && a >= b )
        &&
        ( a % b > 0 )
        &&
        TRUE

    <===>
        (a > 0 && b > 0 && a >= b)
        &&
        ( a % b > 0 )
        &&
        ( gcd(a,b) == gcd(a,b) )

    <===> + APPLY RULE iv. as b > 0 +
        (a > 0 && b > 0 && a >= b)
        &&
        (a % b > 0)
        &&
        ( gcd(b,(a % b)) == gcd(a,b) )

    <===> + a >= b && b > 0 SO a > 0 +
          + a % b != 0 && a % b > 0 SO a % b > 0 +
        (a >= b) && (a % b) != 0 )
        &&
        ( b > 0 && a % b > 0 )
        &&
        ( gcd(b,(a % b)) == gcd(a,b) )

*/ // CASE a >= b && a % b != 0
```

```
// CASE 1

// FINALLY COLELCTING IT ALL TOGETHER
// AND INDEPENDANTLY PROVING EACH CASE
// DIVDING LOGIC INTO 3 CASES

/*
    (a >= b) || (
        ( b > 0 && a > 0 ) &&
        ( gcd(b,a) == gcd(a,b) )
    )
    &&
    ( a < b || ( (a % b) != 0 ) ) || (
        b == gcd(a,b)
    )
    &&
    ( a < b || (a % b) == 0 ) || (
        ( b > 0 && a % b > 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )
    )
*/
/*
    (a < b) ==> (
        ( b > 0 && a > 0 ) &&
        ( gcd(b,a) == gcd(a,b) )
    )
    &&
    ( a >= b && ( (a % b) == 0 ) ) ==> (
        b == gcd(a,b)
    )
    &&
    ( a >= b && (a % b) != 0 ) ==> (
        ( b > 0 && a % b > 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )
    )
*/
/*
    (a < b) ==> (
        ( b > 0 && a > 0 ) &&
        ( gcd(b,a) == gcd(a,b) )
    )
    &&
    ( !(a < b) && ( (a % b) == 0 ) ) ==> (
        b == gcd(a,b)
    )
    &&
    ( !(a < b) && !( (a % b) == 0 ) ) ==> (
        ( b > 0 && a % b > 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )
    )
*/
```

```
    if (a < b)
    {
/*
        ( b > 0 && a > 0 ) &&
        ( gcd(b,a) == gcd(a,b) )

        + One Point Rule +
        ( b > 0 && a > 0 ) &&
            forall r' ::
                ( r' == gcd(b,a) )
        ==> ( r' == gcd(a,b) )

       ( a > 0 && b > 0 )[a,b\b,a] &&
            forall r' ::
                ( r == gcd(a,b) )[a,b,r\b,a,r']
        ==> ( r == gcd(a,b) )[r\r']

        wp(r := M(E1,E2), Q)
            <===>
        P[a,b\E1,E2] &&
            forall r' ::
                R[a,b,r\E1,E2,r']
        ==> Q[r\r']

        + Method Rule +
 */
        r := GCD1(b, a);
        // r == gcd(a,b)
    }
```

```
    else if (a % b == 0)
    {
//      b == gcd(a,b)
        r := b; // same as gcd(b,a%b) == gcd(b,0) == b
//      r == gcd(a,b)
    }
    else
    {
/*
        ( b > 0 && a % b > 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )

        + One Point Rule +
        ( b > 0 && a % b > 0 ) &&
            forall r' ::
                ( r' == gcd(b,(a % b)) )
        ==> ( r' == gcd(a,b) )

       ( a > 0 && b > 0 )[a,b\b,a%b] &&
            forall r' ::
                ( r == gcd(a,b) )[a,b,r\b,(a % b),r']
        ==> ( r == gcd(a,b) )[r\r']

        wp(r := M(E1,E2), Q)
            <===>
        P[a,b\E1,E2] &&
            forall r' ::
                R[a,b,r\E1,E2,r']
        ==> Q[r\r']

        + Method Rule +
 */
        r := GCD1(b, a % b);
//      r == gcd(a,b)
    }
}
// r == gcd(a,b)
```

```
/*
    RULE
    v. ( (a % b) == 0 && b > 0 ) ==> b == gcd(a,b)
    v. ( (a % b) != 0 || b <= 0 ) || b == gcd(a,b)
*/

/* // END PROOF RULE v. HERE

    ( (a % b) == 0 && b > 0 ) ==> b == gcd(a,b)
+ FINALLY ADD THE ASSUMES

    b == b
+ APPLY RULE i.

    b == gcd(b, 0)
    b == gcd(b, a % b)
    F || b == gcd(b, a % b)
+ ASSERT a % b == 0

    a % b != 0 || b == gcd(b, a % b)
    a % b == 0 ==> b == gcd(b,a % b)
+ APPLY RULE iv.

    a % b == 0 ==> b == gcd(a,b)
+ ASSUME b > 0

    ( (a % b) == 0 ) ==> b == gcd(a,b)
+ DERIVING RULE v. (for a >= 0 && b >= 0)

*/ // START PROOF v. HERE
```

END GCD1

GCD2

```
method GCD2(a: int, b: int) returns (r: int)
requires a >= 0 && b >= 0
decreases b
//ensures gcd(a,b)
{
/*
    + APPLY ASSUMPTION a >= 0 as we applied rules that require a >= 0 +
    a >= 0
    &&
    b >= 0

    b == 0 && TRUE
    b > 0

    b == 0 && a == a
    &&
    b > 0

    + RULE i. (assume a >= 0 for rules to apply) +
    b == 0 && a == gcd(a,0)
    &&
    b > 0

    + JUSTIFICATION b > 0 means division by 0 averted in % +
    b == 0 && a == gcd(a,b)
    &&
    b > 0
        && TRUE
        && TRUE

    + IN DAFNY y != 0 ==> (x % y) >= 0 +
    + RULE iv. (assume a >= 0 for rules to apply) +
    b == 0 && a == gcd(a,b)
    &&
    b > 0
        && a % b >= 0
        && gcd(a,b) == gcd(a,b)

    b == 0 && a == gcd(a,b)
    &&
    b > 0
        && a % b >= 0
        && gcd(b,a%b) == gcd(a,b)

    b == 0 && a == gcd(a,b)
    &&
    b > 0 && b >= 0
        && a % b >= 0
        && gcd(b,a%b) == gcd(a,b)
```

```
    + REMOVE b < 0 as +
    + cannot apply rules involving gcd(a,b) as it is holds for a:nat, b:nat +
    ( b < 0 || b > 0 ) || a == gcd(a,b)
    &&
    b == 0 || ( b >= 0 && a % b >= 0 ) &&
              ( gcd(b,(a % b)) == gcd(a,b) )

    b != 0 || a == gcd(a,b)
    &&
    b == 0 || ( b >= 0 && a % b >= 0 ) &&
              ( gcd(b,(a % b)) == gcd(a,b) )

    b == 0 ==> a == gcd(a,b)
    &&
    b != 0 ==>
        ( b >= 0 && a % b >= 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )
*/
    if (b == 0) {
        // a == gcd(a,b)
        r := a;
        // r == gcd(a,b)
    } else {
/*
        ( b >= 0 && a % b >= 0 ) &&
        ( gcd(b,(a % b)) == gcd(a,b) )

        + One Point Rule +
        ( b >= 0 && a % b >= 0 ) &&
            forall r' ::
                ( r' == gcd(b,(a % b)) )
        ==> ( r' == gcd(a,b) )

       ( a >= 0 && b >= 0 )[a,b\b,a%b] &&
            forall r' ::
                ( r == gcd(a,b) )[a,b,r\b,(a % b),r']
        ==> ( r == gcd(a,b) )[r\r']

        wp(r := M(E1,E2), Q)
            <===>
        P[a,b\E1,E2] &&
            forall r' ::
                R[a,b,r\E1,E2,r']
        ==> Q[r\r']

        + Method Rule +
 */
        r := GCD2(b, a % b);
        // r == gcd(a,b)
    }
    // r == gcd(a,b)
}
// r == gcd(a,b)
```

END GCD2