```
method GCD1(a: int, b: int) returns (r: int)
    requires a > 0 && b > 0 && a != b
    ensures r == gcd(a, b)
    decreases a % b
{
    {a > 0 && b > 0 && a != b}
    {(0 < a < b) || (0 < b < a)}
    {(0 < a < b) || false || false || 0 < b < a || false}
    {(0 < a < b) || (0 < a < b && b == gcd(a, b)) || false || 0 < b < a || (b ==
gcd(a, b) && 0 < b < a)}
    {(0 < a < b && a % b != 0) || (0 < a < b && b == gcd(a, b)) || (0 < a < b && 0
< b < a) || (a % b != 0 && 0 < b < a) || (b == gcd(a, b) && 0 < b < a)}
    {(0 < a < b) && (a % b != 0 || b == gcd(a, b)) && (0 < b < a)}
    {(a > b || b > 0 && a > 0) && true && a < b || a % b == 0 || (b > 0 && a % b >
0)}
    {(a < b ==> b > 0 && a > 0) && (a % b == 0 ==> b == gcd(a, b)) && ((a > b && a
% b != 0) ==> b > 0 && a % b > 0)}
    if a < b {
        {b > 0 && a > 0}
        {b > 0 && a > 0 && true}
        {b > 0 && a > 0 && gcd(b, a) == gcd(a, b)}
        {b > 0 && a > 0 && forall r' :: r' == gcd(b, a) ==> r' == gcd(a, b)}
        r := GCD1(b, a);
        {r == gcd(a, b)}
    } else if (a % b == 0) {
        {b == gcd(a, b)}
        r := b;
        {r == gcd(a, b)}
    } else {
        {b > 0 && a % b > 0}
        {b > 0 && a % b > 0 && true}
        {b > 0 && a % b > 0 && gcd(b, a % b) == gcd(a, b)}
        {b > 0 && a % b > 0 && forall r' :: r' == gcd(b, a % b) ==> r' == gcd(a,
b)}
        {true && forall r' :: r' == gcd(b, a % b) ==> r' == gcd(a, b)}
        r := GCD1(b, a % b);
        {r == gcd(a, b)}
    }
    {r == gcd(a, b)}
}
{r == gcd(a, b)}
```

```
method GCD2(a: int, b: int) returns (r: int)
    requires a >= 0 && b >= 0
    ensures r == gcd(a, b)
    decreases b + 1
{
    {a >= 0 && b >= 0}                              (from gcd rules assumption)
    {true}
    {true && true}                                  (used rule i and tautology
condition)
    {b == 0 ==> a == gcd(a, b) && b != 0 ==> true}
    if b == 0 {
        {a == gcd(a, b)}
        r := a;
        {r == gcd(a, b)}
    } else {
        {true}
        {gcd(b, a % b) == gcd(a, b)}
        {true && forall r' :: r' == gcd(b, a % b) ==> r' == gcd(a, b)}
        r := GCD2(b, a % b);
        {r == gcd(a, b)}
    }
    {r == gcd(a, b)}
}
{r == gcd(a, b)}
```