1.
2.
3. function dividePolynomials(z, y):

```
function dividePolynomials(z, y):
    if y == 0: # can't divide by 0
        return NaN

    a = 0  # future quotient
    b = z  # remainder initialized

    while degree(b) >= degree(y):
        c = b[degree(b)] / y[degree(y)]  # the division
        e = degree(b) - degree(y)  # calculate the difference in exponents to make $x^e$

        d = c * $x^e$  # singular term created by the division
        a = a + d  # quotient equation adding next term
        b = b - (d * y)  # create the new reminder that i going to be used in the futured

    return a
```

Loop invariant

z = (y * a) + b ∧ degree(b) >= degree(y)

Base case

a = 0

b = z

z = (y * a) + b ∧ degree(b) >= degree(y)

z=(y · a)+b

z=(y · 0)+z

z=z

degree(b) >= degree(y) has to be true in order for the loop to even run therefore the base case is true

K and K+1 case

z = (y * a_k) + b_k ∧ degree(b_k) >= degree(y)

z = (y * a_{k+1}) + b_{k+1} ∧ degree(b_{k+1}) >= degree(y)

z = (y * a_{k+1}) + b_{k+1}

z = (y * a_k + d) + b_k - (d*y)

z = ya_k+ yd + b_k - dy

z = ya_k+ b_k = z = (y * a_k) + b_k

For the loop to run, degree(b_k) >= degree(y) and degree(b_{k+1}) >= degree(y) have to be true therefore this part is true.

K and K+1 were valid therefore this part of the loop invariant is true.