```
#/**
#Performs polynomial long division. Reads in the coefficients of the dividend and divisor from PolynomialLongDivision::
    g_DividendCoefficients and PolynomialLongDivision::g_DivisorCoefficients, respectively. Writes out the coefficients of the
    quotient and remainder to PolynomialLongDivision::g_QuotientCoefficients and PolynomialLongDivision::
    g_RemainderCoefficients, respectively. The divisor must be non-zero and have the same number of values as the dividend.
#Source of algorithm: https://en.wikipedia.org/w/index.php?title=Polynomial_long_division#Pseudocode
#*/

#IF d ≠ 0
#       RETURN ERROR
#ENDIF
Set Variable [ $n; Value:PolynomialLongDivision::g_DividendCoefficients ]
Set Variable [ $d; Value:PolynomialLongDivision::g_DivisorCoefficients ]

If [ PolynomialIsZero ( $d ) ]
    Show Custom Dialog [ Title: "Error"; Message: "Divisor must be non-zero"; Default Button: “OK”, Commit: “Yes” ]
    Exit Script [  ]
End If

Set Variable [ $nValueCount; Value:ValueCount ( $n ) ]

#ensure polynomials have same number of values to facilitate computations
If [ ValueCount ( $d ) ≠ $nValueCount ]
    Show Custom Dialog [ Title: "Error"; Message: "Divisor must have same number of values as dividend"; Default Button: “OK”,
        Commit: “Yes” ]
    Exit Script [  ]
End If

#q := 0
Set Variable [ $q; Value:"0" ]
Set Variable [ $valueNumber; Value:2 ]

Loop
    Exit Loop If [ $valueNumber > $nValueCount ]
    Set Variable [ $q; Value:$q & "¶0" ]
    Set Variable [ $valueNumber; Value:$valueNumber + 1 ]
End Loop

#r := n
Set Variable [ $r; Value:$n ]

#WHILE r ≠ 0 AND degree(r) ≥ degree(d)
Loop
    Perform Script [ “GetPolynomialDegree”; Parameter: $r ]
    Set Variable [ $rDegree; Value:Get ( ScriptResult ) ]
    Perform Script [ “GetPolynomialDegree”; Parameter: $d ]
    Exit Loop If [ PolynomialIsZero ( $r ) or $rDegree < Get ( ScriptResult ) ]

    #t := lead(r) / lead(d)
    Perform Script [ “GetPolynomialLead”; Parameter: $r ]
    Set Variable [ $rLead; Value:Get ( ScriptResult ) ]
    Perform Script [ “GetPolynomialLead”; Parameter: $d ]
    Set Variable [ $dLead; Value:Get ( ScriptResult ) ]
    Set Variable [ $tLeadCoefficient; Value:GetValue ( $rLead ; 1 ) / GetValue ( $dLead ; 1 ) ]
    Set Variable [ $tDegree; Value:GetValue ( $rLead ; 2 ) - GetValue ( $dLead ; 2 ) ]
```

```
    #q := q + t
    Set Variable [ $q; Value:Let (
            [ numberOfLeftValues = ValueCount ( $q ) - ( $tDegree + 1 ) ;
            rightValues = TrimTrailingNewline (
                RightValues ( $q ; $tDegree )
            ) ] ;
            LeftValues ( $q ; numberOfLeftValues ) &
            ( GetValue ( $q ; numberOfLeftValues + 1 ) + $tLeadCoefficient ) &
            Case (
                IsEmpty ( rightValues ) ; "" ; "¶" & rightValues
            )
        ) ]

    #r := r - t * d
    #multiply t by d
    #compute tLeadCoefficient * d
    Set Variable [ $tTimesD; Value:$d ]
    Set Variable [ $valueNumber; Value:1 ]

    Loop
        Exit Loop If [ $valueNumber > $nValueCount ]
        Set Variable [ $tTimesD; Value:Let (
            [rightValues = TrimTrailingNewline (
                RightValues ( $tTimesD ; $nValueCount - $valueNumber
                )
            )] ;
            LeftValues ( $tTimesD ; $valueNumber - 1 ) &
          ( GetValue ( $tTimesD ; $valueNumber ) * $tLeadCoefficient ) &
            Case (
                IsEmpty ( rightValues ) ; "" ; "¶" & rightValues
            )
          ) ]
        Set Variable [ $valueNumber; Value:$valueNumber + 1 ]
    End Loop

    #compute (tLeadCoefficient * d) * x^tDegree
    Set Variable [ $tTimesD; Value:RightValues ( $tTimesD ; $nValueCount - $tDegree ) &
        TrimTrailingNewline (
            LeftValues ( $tTimesD ; $tDegree )
        ) ]

    #subtract (t * d) from r
    Set Variable [ $valueNumber; Value:1 ]

    Loop
        Exit Loop If [ $valueNumber > $nValueCount ]
        Set Variable [ $r; Value:Let (
            [ rightValues = TrimTrailingNewline (
                RightValues ( $r ; $nValueCount - $valueNumber )
            )
            ] ;
            LeftValues ( $r ; $valueNumber - 1 ) &
            ( GetValue ( $r ; $valueNumber ) - GetValue ( $tTimesD ; $valueNumber ) ) &
            Case (
                IsEmpty ( rightValues ) ; "" ; "¶" & rightValues
            )
          ) ]
        Set Variable [ $valueNumber; Value:$valueNumber + 1 ]
    End Loop
End Loop
#ENDWHILE
```

#RETURN (q, r)
Freeze Window
Set Field [PolynomialLongDivision::g_QuotientCoefficients; $q]
Set Field [PolynomialLongDivision::g_RemainderCoefficients; $r]
Commit Records/Requests
[No dialog]