Pseudocode

Option L — quartilesCalculation()

Purpose: Calculates the first quartile (Q1), median (Q2), and third quartile (Q3) of the dataset and determines whether Q1 and Q3 can be calculated (known) or not.
Precondition: Dataset must have at least 2 values.
Postcondition: Returns Q1, Q2, Q3 and whether Q1 and Q3 are known or unknown (via QuartileValues struct).
Time Complexity:
O(n)

```
FUNCTION quartilesCalculation() RETURNS QuartileValues

    n = size of dataset

    //takes care of invalid dataset size
    IF n == 0 THEN
        PRINT "Error: Dataset is empty"
        RETURN
    END IF

    IF n == 1 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN
    END IF

    //Q2 is the same as the median of whole data set
    Q2 = median(dataset)

    //start of finding Q1
    Q1 = 0
    q1Known = FALSE             //so we can print "unknown"
    lowerHalfSize = n / 2

    IF lowerHalfSize >= 2 THEN
        q1Known = TRUE
        IF lowerHalfSize is odd THEN
            Q1 = dataset[lowerHalfSize / 2]
        ELSE
```

```
            mid1 = dataset[(lowerHalfSize / 2) - 1]
            mid2 = dataset[lowerHalfSize / 2]
            Q1 = (mid1 + mid2) / 2.0
        END IF
    END IF

    //start of finding Q3
    Q3 = 0
    q3Known = FALSE

    //find where the upper half starts
    IF n is even THEN
        upperStart = n / 2
    ELSE
        upperStart = (n / 2) + 1
    END IF

    upperHalfSize = n - upperStart

    IF upperHalfSize >= 2 THEN
        q3Known = TRUE
        IF upperHalfSize is odd THEN
            Q3 = dataset[upperStart + (upperHalfSize / 2)]
        ELSE
            mid1 = dataset[upperStart + (upperHalfSize / 2) - 1]
            mid2 = dataset[upperStart + (upperHalfSize / 2)]
            Q3 = (mid1 + mid2) / 2.0
        END IF
    END IF

    RETURN (Q1, Q2, Q3, q1Known, q3Known)
END FUNCTION
```

Option M — interquartile()

Purpose: Calculates and returns the Interquartile Range (IQR = Q3 – Q1), a measure of spread for the middle 50% of the data.

Precondition: Dataset must have at least 4 values.

Postcondition: Returns IQR as a double.

Time Complexity: O(n)

```
FUNCTION interquartile() RETURNS double

    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN
    END IF

    q = quartilesCalculation()
    IQR = q.Q3 - q.Q1

    PRINT "Interquartile Range = ", IQR

    RETURN IQR
END FUNCTION
```

Option N — outliers()

Purpose: Identifies data points considered outliers (values outside 1.5 × IQR from Q1 or Q3).
Precondition: Dataset must have at least 2 values; Q1 and Q3 must be known.
Postcondition: Returns a DynamicArray containing outlier values (if any).
Time Complexity: O(n)

```
FUNCTION outliers() RETURNS DynamicArray<double>

    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN empty array
    END IF

    q = quartilesCalculation()
    IQR = q.Q3 - q.Q1

    lowerFence = q.Q1 - (1.5 * IQR)
    upperFence = q.Q3 + (1.5 * IQR)

    outlierValues = empty DynamicArray
    FOR i = 0 TO n - 1
        IF dataset[i] < lowerFence OR dataset[i] > upperFence THEN
            outlierValues.append(dataset[i])
        END IF
    END FOR

    IF outlierValues.size() == 0 THEN
        PRINT "Outliers = None"
    ELSE
        PRINT "Outliers = ", all values in outlierValues
    END IF

    RETURN outlierValues
END FUNCTION
```

Option O — sumOfSquares()

Purpose: Calculates the sum of squared deviations of data values from the mean, a measure of total variation in the dataset.
Precondition: Dataset must have at least 2 values.
Postcondition: Returns sum of squares.
Time Complexity: O(n)

```
FUNCTION sumOfSquares() RETURNS double

    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN 0
    END IF

    meanValue = mean(dataset)
    total = 0

    FOR i = 0 TO n - 1
        total = total + (dataset[i] - meanValue)^2
    END FOR

    PRINT "Sum of Squares = ", total
    RETURN total
END FUNCTION
```

Option P — meanAbsoluteDeviation()

Purpose: Calculates and returns the Mean Absolute Deviation (MAD), the average of absolute deviations of data values from the mean.
Precondition: Dataset must have at least 2 values.
Postcondition: Returns MAD.
Time Complexity: O(n)

```
FUNCTION meanAbsoluteDeviation() RETURNS double

    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN 0
    END IF

    meanValue = mean(dataset)
    sum = 0

    FOR i = 0 TO n - 1
        sum = sum + ABS(dataset[i] - meanValue)
    END FOR

    MAD = sum / n
    PRINT "Mean Absolute Deviation = ", MAD

    RETURN MAD
END FUNCTION
```

Option Q — rootMeanSquare()

Purpose: Calculates and returns the Root Mean Square (RMS), the square root of the mean of squared values.
Precondition: Dataset must have at least 2 values.
Postcondition: Returns RMS.
Time Complexity: O(n)

```
FUNCTION rootMeanSquare() RETURNS double
```

```
    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN 0
    END IF

    total = 0
    FOR i = 0 TO n - 1
        total = total + (dataset[i]^2)
    END FOR

    RMS = SQRT(total / n)
    PRINT "Root Mean Square = ", RMS

    RETURN RMS
END FUNCTION
```

Option R — standardErrorMean()

Purpose: Calculates and returns the Standard Error of the Mean (SEM), which estimates how far the sample mean is expected to vary from the true population mean.
Precondition: Dataset must have at least 2 values.
Postcondition: Returns SEM as a double.
Time Complexity: O(n)

```
FUNCTION standardErrorMean() RETURNS double

    n = size of dataset
    IF n < 2 THEN
        PRINT "Error: Requires at least 2 data values"
        RETURN 0
    END IF

    s = standardDeviation(dataset)
    SEM = s / SQRT(n)

    PRINT "Standard Error of the Mean = ", SEM

    RETURN SEM
END FUNCTION
```

Option Y and Z — dataDisplay()

Purpose: Displays all statistical results of the dataset to an output stream (console or text file), including measures of center, spread, and shape.

Precondition: Dataset must not be empty.

Postcondition: Displays or writes formatted statistical results.

Time Complexity: O(n)

```
FUNCTION dataDisplay(outputStream)

    //Handles both console and file output
    n = size of dataset

    IF n == 0 THEN
        PRINT "Error: Dataset is empty"
        RETURN
    END IF

    SORT dataset

    PRINT "Minimum = ", minimum(dataset)
    PRINT "Maximum = ", maximum(dataset)
    PRINT "Range = ", range(dataset)
    PRINT "Size = ", n
    PRINT "Sum = ", sum(dataset)
    PRINT "Mean = ", mean(dataset)
    PRINT "Median = ", median(dataset)
    PRINT "Mode = ", mode(dataset)
    PRINT "Standard Deviation = ", standardDeviation(dataset)
    PRINT "Variance = ", variance(dataset)
    PRINT "Midrange = ", midrange(dataset)

    q = quartilesCalculation()
    PRINT "Quartile 1 (Q1) = ", q.Q1
    PRINT "Quartile 2 (Q2) = ", q.Q2
    PRINT "Quartile 3 (Q3) = ", q.Q3

    IQR = interquartile()
    PRINT "Interquartile Range = ", IQR

    outliersList = outliers()
    PRINT "Outliers = ", (if none then "None" else outliersList)

    PRINT "Sum of Squares = ", sumOfSquares()
    PRINT "Mean Absolute Deviation = ", meanAbsoluteDeviation()
    PRINT "Root Mean Square = ", rootMeanSquare()
    PRINT "Standard Error of the Mean = ", standardErrorMean()
    PRINT "Skewness = ", skewness()
    PRINT "Kurtosis = ", kurtosis()
    PRINT "Kurtosis Excess = ", kurtosisExcess()
    PRINT "Coefficient of Variation = ", coefficientOfVariation()
    PRINT "Relative Standard Deviation = ", relativeStandardDeviation()

    CALL displayFrequencyTable(outputStream)

END FUNCTION
```