CAD Pseudo Code

```
*********************************************************
        CAD:
                    A function that finds the anomalies of a time series of random deviates

        Calls:
                    anomaly_finder

        Called by:
                    None

        Input Parameters:
                    time_series - the time series being searched for the anomalies
                    delta - the value to be added to the value to the CUSUM parameter k; its
                          default value is 3
                    lambda - the minimum length of the anomalous subsequence the code should
                          detect; the default is 5
                    type - the type of the anomaly "upper" or "lower"
                    number_of_windows - the number of windows used
                    step_size - the distance between two consecutive windows
                    training_set_window_length - the starting length of the training set searched
                          for within each window
                    training_set_step_size - the size of the shift by which the
                          training_set_windows are moved down the sequence.

        Returns:
                    anomaly_indices - The indices of the anomalies
        ************************************************************
```

```
FUNCTION  CAD(time_series, delta, lambda, type,
          number_of_windows, step_size, training_set_window_length,
          training_set_step_size)

          n <-LEN(time_series)
          window_length <- n - (number_of_windows*step_size) + 1
          INIT anomaly_indices <- NULL

          FOR i FROM 1 TO (number_of_windows*step_size) BY  step_size

                temp_var <- time_series[i TO (i+window_length -1)]
                temp_indices <- anomaly_finder(temp_var, delta, lambda,
                                type, i, step_size,
                                training_set_window_length,
                                training_set_step_size)
```

```
                anomaly_indices <- CONCATENATE(anomaly_indices,
                                    temp_indices)

        ENDFOR

        RETURN(anomaly_indices)

ENDFUNCTION
```

```
**********************************************************
        anomaly_finder:
                A function that finds the anomalies for a specific window
                of a time series
        Calls:
                training_subsequence_finder
                evaluate_CUSUM_results

        Called by:
                CAD

        Input Parameters:
                subsqnce  - a window length subsequence of the original
                        time series.
                delta - the value to be added to the value to the CUSUM parameter k; its
                        default value is 3
                lambda - the minimum length of the anomalous subsequence the code should
                        detect; the default is 5
                type - the type of the anomaly "upper" or "lower"
                indx - the index of the first element of sbsqnce within time_series.
                step_size - the distance between two consecutive windows
                training_set_window_length - the starting length of the training set searched
                        for within each window
                training_set_step_size - the size of the shift by which the
                        training_set_windows are moved down the sequence.

        Returns:
                global_indices - The indices of the anomalies within the original time_series.
        ************************************************************
```

```
FUNCTION anomaly_finder(subsqnce, delta, lambda, type, indx, step_size,
                        training_window_length, training_set_step_size)

        training_set_object <- training_subsequence_finder(subsqnce,
                                training_window_length, training_set_step_size)

        IF  LEN(training_set_object[1])== 0 THEN
                        WARNING("No training set was found.")
                        RETURN(NULL)
        ELSE
                x_bar <- MEAN(training_set_object[1])
```

```
                sigma <- STANDARD_DEVIATION(training_set_object[1])
                k <- training_set_object[2] + delta
                H <- trainig_set_object[3]
                local_indices <- evaluate_CUSUM_results(subseqnce,
                     x_bar,sigma,H,k,type,lambda)
                IF LEN(local_indices)!= 0 THEN
                     global_indices <- local_indices + indx -1
                ELSE
                     global_indices <- NULL
                ENDIF
                RETURN(global_indices)


          ENDIF

ENDFUNCTION



**********************************************************
        training_subsequence_finder:
                A function that searches for a subsequence of a time series that is both (close
                to) normal and in statistical control.  This subsequence must have a minimal
                length of 30 and maximal length of training_set_window_length.
        Calls:
                normality_finder
                parameter_finder

        Called by:
                anomaly_finder

        Input Parameters:
                x - a time series; it must have length greater than 30.
\
                training_set_window_length - the starting length of the training set searched
                      for within each window
                training_set_step_size - the size of the shift by which the
                      training_set_windows are moved down the sequence.

        Returns:
                training_set_object - a list/object of length 3, where
                      training_set_object[1] - the subsequence of x that is to be used
                            as the training data set
                      training_set_object[2] - the CUSUM k value of training_set_object[1]
                      training_set_object[3] - the CUSUM H value of training_set_object[1]

**********************************************************


FUNCTION training_subsequence_finder(x, training_set_window_length,
     training_set_step_size)

     INIT segment_found <- FALSE
     INIT cant_be_done <- FALSE
```

```
INIT first_time_through <- TRUE
INIT parameters_plus <- NULL
n <- LEN(x)

WHILE (NOT segment_found) AND ( NOT cant_be_done)

     FOR i FROM 1 TO (n - training_set_window_length +1) BY
          training_set_step_size

          temp <- x[i TO (i+training_set_window_length-1)]
          temp_obj <- normality_finder(temp)

          IF temp_obj[1]> 0.05 THEN
               p_value <- temp_obj[1]
               skew <- temp_obj[2]
               kurt <- temp_obj[3]
               cusum_parameters_obj <-  parameter_finder(temp)
               the_J <- ABS(skew) + ABS(3 - kurt)+ ABS(1-p_value)+
                    (100*cusum_parameters_obj[1]) +
                    (100*cusum_parameters_obj[2])
               dummy_var <- [temp, p_value, skew,
                    cusum_parameters_obj[1],
                    cusum_parameters_obj[2],the_J]

               IF first_time_through THEN
                    parameters_plus <- dummy_var
                    first_time_through <- FALSE

               ELSE
                    IF parameters_plus[6] > the_J
                         parameters_plus <- dummy_var
                    ENDIF
               ENDIF
          ENDIF
     ENDFOR

     IF parameters_plus != NULL THEN
          segment_found <- TRUE
     ELSE
          IF training_set_window_length > 30 THEN
               training_set_window_length <-
                    training_set_window_length -1
          ELSE
               cant_be_done <- TRUE
          ENDIF
     ENDIF
```

```
     ENDWHILE

     IF cant_be_done THEN
          return_obj <- NULL
     ELSE
          return_obj <- [parameters_plus[1], parameters_plus[4],
               parameters_plus[5]]
     ENDIF

     RETURN(return_obj)
ENDFUNCTION
```