1.1.28 Remove duplicates. Modify the test client in BinarySearch to remove any duplicate keys in the whitelist after the sort.

|  |
| --- |
| FUNCTION removeDuplicates(array)  DECLARE uniqueList AS EMPTY LIST  DECLARE previousValue AS NULL  FOR EACH value IN array DO  IF value != previousValue THEN  APPEND value TO uniqueList  ENDIF  previousValue = value  ENDFOR  RETURN uniqueList  END FUNCTION |

1.1.29 Equal keys. Add to BinarySearch a static method rank() that takes a key and a sorted array of int values (some of which may be equal) as arguments and returns the number of elements that are smaller than the key and a similar method count() that returns the number of elements equal to the key. Note : If i and j are the values returned by rank(key, a) and count(key, a) respectively, then a[i..i+j-1] are the values in the array that are equal to key.

|  |
| --- |
| ***FUNCTION rank(key, array)***  DECLARE lo AS INTEGER = 0  DECLARE hi AS INTEGER = LENGTH(array) - 1  WHILE lo <= hi DO  DECLARE mid AS INTEGER = (lo + hi) / 2    IF key < array[mid] THEN  SET hi = mid - 1  ELSE IF key > array[mid] THEN  SET lo = mid + 1  ELSE  // Found key, return the number of elements smaller than key  RETURN mid // We return the index of the first occurrence of key  ENDIF  ENDWHILE    RETURN lo // Return the position where key would be inserted  END FUNCTION  ***FUNCTION count(key, array)***  DECLARE index AS INTEGER = rank(key, array)    IF index == -1 THEN  RETURN 0 // Key not found  ENDIF  DECLARE count AS INTEGER = 0  DECLARE i AS INTEGER = index    **// Count elements equal to key on the left side**  WHILE i >= 0 AND array[i] == key DO  INCREMENT count  DECREMENT i  ENDWHILE    **// Count elements equal to key on the right side**  i = index + 1  WHILE i < LENGTH(array) AND array[i] == key DO  INCREMENT count  INCREMENT i  ENDWHILE    RETURN count  END FUNCTION |

1.1.30 Array exercise. Write a code fragment that creates an N-by-N boolean array a[][] such that a[i][j] is true if i and j are relatively prime (have no common factors), and false otherwise.

|  |
| --- |
| **FUNCTION createRelativelyPrimeArray(N)**  *// Step 1: Initialize the N x N boolean array*  DECLARE a AS ARRAY OF BOOLEAN[N][N]  *// Step 2: Fill the array*  FOR i FROM 0 TO N-1 DO  FOR j FROM 0 TO N-1 DO  *// Step 3: Check if i and j are relatively prime*  IF gcd(i, j) == 1 THEN  a[i][j] = TRUE  ELSE  a[i][j] = FALSE  ENDIF  ENDFOR  ENDFOR  RETURN a // Return the populated boolean array  END FUNCTION  **FUNCTION gcd(x, y)**  WHILE y != 0 DO  DECLARE temp AS INTEGER = y  y = x MOD y  x = temp  ENDWHILE  RETURN x // Return the GCD of x and y  END FUNCTION |

1.1.31 Random connections. Write a program that takes as command-line arguments an integer N and a double value p (between 0 and 1), plots N equally spaced dots of size .05 on the circumference of a circle, and then, with probability p for each pair of points, draws a gray line connecting them.

|  |
| --- |
| FUNCTION RandomConnections  FUNCTION main(args):  IF LENGTH(args) != 2 THEN  PRINT "Usage: RandomConnections <number\_of\_points> <probability>"  RETURN  N = CONVERT args[0] TO INTEGER  p = CONVERT args[1] TO FLOAT  IF N <= 0 THEN  PRINT "Error: N must be positive."  RETURN  ENDIF  IF p < 0 OR p > 1 THEN  PRINT "Error: Probability p must be between 0 and 1."  RETURN  ENDIF  INITIALIZE WINDOW with title "Random Connections"  CREATE RandomConnections instance with parameters N and p  SET WINDOW properties (size, close operation)  DISPLAY WINDOW  END FUNCTION  CLASS RandomConnections:    CONSTRUCTOR(N, p):  SET this.N = N  SET this.p = p  END CONSTRUCTOR  FUNCTION paintComponent(graphics):  CALL super.paintComponent(graphics)  g2d = CAST graphics TO 2D Graphics  width = GET width of drawing area  height = GET height of drawing area  centerX = width / 2  centerY = height / 2  radius = MIN(width, height) / 2.5  // Draw the dots  FOR i FROM 0 TO N - 1 DO:  angle = 2 \* PI \* i / N  x = centerX + radius \* COS(angle)  y = centerY + radius \* SIN(angle)  DRAW dot at (x, y) with a predefined size  // Draw lines based on probability p  RANDOM randomGenerator  FOR i FROM 0 TO N - 1 DO:  FOR j FROM i + 1 TO N - 1 DO:  IF randomGenerator.NEXT\_FLOAT() < p THEN:  angle1 = 2 \* PI \* i / N  angle2 = 2 \* PI \* j / N  x1 = centerX + radius \* COS(angle1)  y1 = centerY + radius \* SIN(angle1)  x2 = centerX + radius \* COS(angle2)  y2 = centerY + radius \* SIN(angle2)  DRAW line from (x1, y1) to (x2, y2)  END FUNCTION  END CLASS  END FUNCTION |

1.1.32 Histogram. Suppose that the standard input stream is a sequence of double values. Write a program that takes an integer N and two double values l and r from the command line and uses StdDraw to plot a histogram of the count of the numbers in the standard input stream that fall in each of the N intervals defined by dividing (l , r) into N equal-sized intervals.

|  |
| --- |
| FUNCTION Histogram  FUNCTION main(args):  // Check if there are exactly 3 command-line arguments  IF LENGTH(args) != 3 THEN  PRINT "Usage: Histogram <N> <l> <r>"  RETURN  // Parse command-line arguments  N = CONVERT args[0] TO INTEGER  l = CONVERT args[1] TO DOUBLE  r = CONVERT args[2] TO DOUBLE  // Validate input  IF N <= 0 OR l >= r THEN  PRINT "Error: N must be positive and l must be less than r."  RETURN  // Create an array to count values in each interval  DECLARE counts AS ARRAY of SIZE N INITIALIZED to 0  intervalSize = (r - l) / N  // Read values from standard input  WHILE there are more values in input DO  value = READ next value from input  // Check if the value is within the range  IF value >= l AND value < r THEN  index = (value - l) / intervalSize  IF index >= N THEN  index = N - 1 // Handle edge case for r  ENDIF  counts[index] = counts[index] + 1  ENDIF  ENDWHILE  // Draw the histogram  SETUP drawing canvas  SET X scale from l to r  SET Y scale from 0 to MAX(counts) + 1  // Draw bars for each interval  FOR i FROM 0 TO N - 1 DO  x = l + i \* intervalSize  width = intervalSize  height = counts[i]  DRAW filled rectangle at (x + width/2, height/2) with width and height  ENDFOR  // Add labels to the histogram  ADD title and labels to the drawing  END FUNCTION  END FUNCTION |

1.1.33 Matrix library. Write a library Matrix that implements the following API:

A screenshot of a computer code

Description automatically generated

Develop a test client that reads values from standard input and tests all the methods.

|  |
| --- |
| **CLASS MatrixLibrary**  *// Function to calculate dot product of two vectors*  **FUNCTION dot(vector x, vector y)**  IF LENGTH(x) != LENGTH(y) THEN  THROW "Vectors must have the same length"  ENDIF    result = 0  FOR i FROM 0 TO LENGTH(x) - 1 DO  result = result + (x[i] \* y[i])  ENDFOR    RETURN result  END FUNCTION  *// Function to multiply two matrices*  **FUNCTION mult(matrix a, matrix b)**  m = NUMBER OF ROWS in a  n = NUMBER OF COLUMNS in a  p = NUMBER OF COLUMNS in b    IF n != NUMBER OF ROWS in b THEN  THROW "Matrices cannot be multiplied"  ENDIF  result = NEW matrix of size m x p  FOR i FROM 0 TO m - 1 DO  FOR j FROM 0 TO p - 1 DO  result[i][j] = 0  FOR k FROM 0 TO n - 1 DO  result[i][j] = result[i][j] + (a[i][k] \* b[k][j])  ENDFOR  ENDFOR  RETURN result  END FUNCTION  *// Function to transpose a matrix*  **FUNCTION transpose(matrix a)**  rows = NUMBER OF ROWS in a  cols = NUMBER OF COLUMNS in a  transposed = NEW matrix of size cols x rows    FOR i FROM 0 TO rows - 1 DO  FOR j FROM 0 TO cols - 1 DO  transposed[j][i] = a[i][j]  ENDFOR  ENDFOR    RETURN transposed  END FUNCTION  *// Function to multiply a matrix by a vector*  **FUNCTION mult(matrix a, vector x)**  m = NUMBER OF ROWS in a  n = NUMBER OF COLUMNS in a    IF n != LENGTH(x) THEN  THROW "Matrix and vector dimensions do not match"  ENDIF  result = NEW vector of size m  FOR i FROM 0 TO m - 1 DO  result[i] = 0  FOR j FROM 0 TO n - 1 DO  result[i] = result[i] + (a[i][j] \* x[j])  ENDFOR  ENDFOR    RETURN result  END FUNCTION  *// Function to multiply a vector by a matrix*  **FUNCTION mult(vector y, matrix a)**  n = LENGTH(y)  m = NUMBER OF ROWS in a  p = NUMBER OF COLUMNS in a    IF n != m THEN  THROW "Vector and matrix dimensions do not match"  ENDIF  result = NEW vector of size p  FOR j FROM 0 TO p - 1 DO  result[j] = 0  FOR i FROM 0 TO n - 1 DO  result[j] = result[j] + (y[i] \* a[i][j])  ENDFOR  ENDFOR    RETURN result  END FUNCTION  END CLASS |

1.1.34 Filtering. Which of the following require saving all the values from standard input (in an array, say), and which could be implemented as a filter using only a fixed number of variables and arrays of fixed size (not dependent on N)? For each, the input comes from standard input and consists of N real numbers between 0 and 1.

■ Print the maximum and minimum numbers.

|  |
| --- |
| **FUNCTION findMaxMin()**  max = 0  min = 1  WHILE there are more numbers to read  number = READ next number  IF number > max THEN  max = number  ENDIF  IF number < min THEN  min = number  ENDIF  ENDWHILE  PRINT max  PRINT min  END FUNCTION |

■ Print the median of the numbers.

|  |
| --- |
| **FUNCTION findMedian()**  DECLARE array numbers of size N  count = 0  WHILE count < N DO  numbers[count] = READ next number  count = count + 1  ENDWHILE  SORT numbers  IF N is odd THEN  median = numbers[N/2]  ELSE  median = (numbers[N/2 - 1] + numbers[N/2]) / 2  ENDIF  PRINT median  END FUNCTION |

■ Print the k th smallest value, for k less than 100.

|  |
| --- |
| **FUNCTION findKthSmallest(k)**  DECLARE array numbers of size N  count = 0  WHILE count < N DO  numbers[count] = READ next number  count = count + 1  ENDWHILE  SORT numbers  PRINT numbers[k-1] // k is 1-based index  END FUNCTION |

■ Print the sum of the squares of the numbers.

|  |
| --- |
| **FUNCTION sumOfSquares()**  sum = 0  WHILE there are more numbers to read  number = READ next number  sum = sum + (number \* number)  ENDWHILE  PRINT sum  END FUNCTION |

■ Print the average of the N numbers.

|  |
| --- |
| **FUNCTION average()**  sum = 0  count = 0  WHILE there are more numbers to read  number = READ next number  sum = sum + number  count = count + 1  ENDWHILE  average = sum / count  PRINT average  END FUNCTION |

■ Print the percentage of numbers greater than the average.

|  |
| --- |
| **FUNCTION percentageAboveAverage()**  DECLARE array numbers of size N  sum = 0  count = 0  WHILE count < N DO  numbers[count] = READ next number  sum = sum + numbers[count]  count = count + 1  ENDWHILE  average = sum / count  greaterCount = 0  FOR i FROM 0 TO count - 1 DO  IF numbers[i] > average THEN  greaterCount = greaterCount + 1  ENDIF  ENDFOR  percentage = (greaterCount / count) \* 100  PRINT percentage  END FUNCTION |

■ Print the N numbers in increasing order.

|  |
| --- |
| **FUNCTION printSortedNumbers()**  DECLARE array numbers of size N  count = 0  WHILE count < N DO  numbers[count] = READ next number  count = count + 1  ENDWHILE  SORT numbers  FOR i FROM 0 TO count - 1 DO  PRINT numbers[i]  ENDFOR  END FUNCTION |

■ Print the N numbers in random order.

|  |
| --- |
| **FUNCTION printRandomOrder()**  DECLARE array numbers of size N  count = 0  WHILE count < N DO  numbers[count] = READ next number  count = count + 1  ENDWHILE  SHUFFLE numbers  FOR i FROM 0 TO count - 1 DO  PRINT numbers[i]  ENDFOR  END FUNCTION |