START Main method

Command line soilComposition String

Command line elementValue String

STORE the value of getElements(soilComposition)

STORE the value of getSamples(elementValue)

PRINT results of searchForLife( value of getElements(soilComposition) )

PRINT results of searchHighestElements( results of getSamples() and getElements() )

PRINT results of searchHighestSample( results of getSamples() and getElements() )

END Main method

START getElements 🡪 public static String[] getElements(String inputElementString)

String[] elementsArray = inputElementString.split(“,”);

RETURN elementsArray

END getElements

START getSamples 🡪 public static double[][] getSamples(String inputSamplesString)

String[] tempArray = inputSamplesString.split(“,|<>”); // | acts as an or

double[][] compositionArray = new String [5][6]

SET index to 0, SET row to 0, SET column to 0

WHILE (index < compositionArray.length && row < compositionArray.length)

compositionArray[row][column] = tempArray[index]

column++

index++

IF (column == compositionArray[row].length)

column 0

row++

END IF

END WHILE

RETURN compositionArray

END getSamples()

START searchForLife() 🡪 public static int[] searchForLife(double [][] samples)

int index = 0; //index used to add to int array we will return at method’s end

int[] tempLifeArray = new int[sample.length]; //array used before we switch it to final array

int count = 0; //count variable used to determine length of final int array we will return

FOR 🡪 for (int row = 0; row < samples.length; row++)

double supportsLife = (8 \* samples[row][0]) + (2 \* samples[row][1]) + samples[row][2] + (4 \* samples[row][3]) + samples[row][4] + (5 \* samples[row]5);

IF (supportsLife >= 300)

int lifeRow = row + 1;

lifeArray[index] = lifeRow;

index++;

count++;

END IF

END FOR

int[] lifeArray = new int[count];

//for loop to switch tempLifeArray elements to lifeArray

FOR 🡪 for (int i = 0; I < lifeArray.length; i++)

lifeArray[i] = tempLifeArray[i];

END FOR

RETURN lifeArray

END searchForLife()

START searchHighestElements 🡪 public static String searchHighestElements(double [][]

samples, String[] elements, int sampleNum)

//set a default highest and second highest. Set vars aside for their indexes also

double highest = samples[sampleNum][0]

int highestRow = sampleNum;

int highestColumn = 0;

double secondHighest = samples[sampleNum][1]

secondHighestRow = sampleNum

sampleHighestColumn = 1

//search the indicated sample

FOR 🡪 for(int column = 0; column < samples[sampleNum].length; column++)

IF (samples[sampleNum][column] > highest)

//move highest and its indexes to secondHighest vars

secondHighest = highest

secondHighestColumn = highestColumn

secondHighestRow = sampleNum

//make current element being traversed the highest

highest = samples[sampleNum][column]

highestRow = sampleNum

highestColumn = column

END IF

ELSE IF (samples[sampleNum][column] > secondHighest && samples[sampleNum][column] < highest)

//make current element second highest

secondHighest = samples[sampleNum][column]

secondHighestColumn = column

secondHighestRow = sampleNum

END ELSE IF

END FOR

//create array to return the string elements that correspond with the numbers

String[] highestElements = {elements[highestColumn], elements[secondHighestColumn]};

RETURN highestElements

END searchHighestElements

START searchHighestSample() 🡪 public static int searchHighestSample(double [][]

samples, String[] elements, String element)

//we will use this for loop to get index of element name in 1d array.

//this index will be used for the 2d array column we will be checking

FOR (int i = 0; i < elements.length; i++)

IF (element.compareTo(elements[i]) == 0)

int elementColumn = i;

END IF

END FOR

//use this for loop to check each row in the column. After checking we will have the highest num

int highestSampleIndex = 0;

FOR 🡪 for (int row = 0; row < samples.length; row++)

IF (samples[row][elementColumn] > samples[highestSampleIndex][elementColumn])

highestSampleIndex = row;

END IF

END FOR

//if highestSampleIndex is 0, we need to indicate that the highest sample’s number is actually

//one in the print statement

RETURN highestSampleIndex;

END searchHighestSample