import java.util.Scanner;

public class NewClass {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int positive = 0, negative = 0, total = 0, count = 0;

float average;

System.out.println("Enter the number: ");

int number;

while((number = input.nextInt()) != 0) {

total += number;

count++;

if(number > 0){

positive++;

} else if(number < 0) {

negative++;

}

} // <-- end loop body.

float average = total / (float) count; // <-- not integer math.

System.out.println("The number of positives is " + positive);

System.out.println("The number of negatives is " + negative);

System.out.println("The total is " + total);

System.out.println("The average is " + average);

\*/

import java.util.Scanner;

public class Exercise\_02 {

public static void main(String[] args) {

// Create a Scanner object

Scanner input = new Scanner(System.in);

// Prompt the user to enter an integer from 1 to 15

System.out.print("Enter the number of lines: ");

int numberOfLines = input.nextInt();

// Display pyramid

for (int rows = 1; rows <= numberOfLines; rows++) {

// Create spaces in each row

for (int s = numberOfLines - rows; s >= 1; s--) {

System.out.print(" ");

}

// Create decending numbers in each row

for (int l = rows; l >= 2; l--) {

System.out.print(l + " ");

}

// Create ascending number in each row

for (int r = 1; r <= rows; r++) {

System.out.print(r + " ");

}

// End line

System.out.println();

}

}

}

\*/

public class Exercise\_03 {

public static void main(String[] args) {

// Sum the series

double sum = 0.0;

for (double n = 1.0; n <= 97.0; n += 2) {

sum += n / (n + 2);

}

System.out.println(

"Series: 1 / 3 + 3 / 5 + 5 / 7 + 7 / 9 + 9 / 11 + 11 / 13 + " +

" ... + 95 / 97 + 97 / 99");

System.out.println("Sum of series: " + sum);

}

}

\*/

public class Exercise\_04 {

public static void main(String[] strings) {

Scanner input = new Scanner(System.in);

System.out.print("Enter an integer: ");

int number = input.nextInt();

String binaryString = "";

while (number >= 1) {

if (number % 2 == 0) {

binaryString = "0" + binaryString ;

} else {

binaryString = "1" + binaryString ;

}

number /= 2;

}

System.out.println(binaryString);

}

}

\*/

import java.util.Scanner;

public class Exercise\_05 {

/\*\* Main Method \*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in); // Create a Scanner

// Prompt the user to enter an integer

System.out.print("Enter an integer: ");

int number = input.nextInt();

// Report whether the integer is a palindrome.

System.out.println(number + (isPalindrome(number) ? " is " : " is not ") +

"a palindrome.");

}

/\*\* Method isPalindrome returns true if number is a palindrome \*/

public static boolean isPalindrome(int number) {

return number == reverse(number) ? true : false;

}

/\*\* Method reverse returns the reversal of an integer \*/

public static int reverse(int number) {

String reverse = ""; // Holds reversed number

String n = number + ""; // Convert number to string

// Reverse string

for (int i = n.length() - 1; i >= 0; i--) {

reverse += n.charAt(i);

}

return Integer.parseInt(reverse); // Return reversed integer

}

}

\*/

public class Exercise\_06 {

/\*\* Main Method \*/

public static void main(String[] args) {

// Display table

double start = 1; // Start series

double end = 901; // End series

System.out.println("\ni m(i) ");

System.out.println("---------------------");

for (double i = start; i <= end; i += 100) {

System.out.printf("%-12.0f", i);

System.out.printf("%-6.4f\n", estimatePI(i));

}

}

/\*\* Method estimatePI \*/

public static double estimatePI(double n) {

double pi = 0; // Set pi to 0

for (double i = 1; i <= n; i ++) {

pi += Math.pow(-1, i +1) / (2 \* i - 1);

}

pi \*= 4;

return pi;

}

}

\*/

public class Exercise\_07 {

public static void main(String[] args) {

int count = 0;

for (int i = 0; count < 100; i++) {

if (isPrime(i) && isPalindrome(i)) {

System.out.printf("%10d", i);

count++;

if (count % 10 == 0 && i != 0) System.out.println();

}

}

}

public static boolean isPrime(long n) {

if (n < 2) return false;

for (int i = 2; i <= n / 2; i++) {

if (n % i == 0) return false;

}

return true;

}

public static long reverse(long number) {

long reverse = 0;

while (number != 0) {

reverse \*= 10; // is ignored first iteration

reverse += number % 10;

number /= 10;

}

return reverse;

}

public static boolean isPalindrome(long number) {

return (number == reverse(number));

}

}

\*/

import java.util.Scanner;

public class Exercise\_08 {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

int[] values = new int[100];

int input;

int count = 0;

System.out.print("Enter the integers between 1 and 100: ");

do

{

input = in.nextInt();

values[count] = input;

count += 1;

}

while (input != 0);

countOccurence(values);

}

public static void countOccurence(int[] list) {

for (int i = 1; i <= 100; i++) {

int count = 0;

for (int j = 0; j < list.length - 1; j++) {

if (list[j] == i)

count += 1;

}

if (count != 0)

System.out.printf("%d occurs %d %s%n",

i, count, count > 1 ? "times" : "time");

}

}

}

\*/

import java.util.Scanner;

public class Exercise\_09 {

/\*\* Main Method \*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

int[] distinctNumbers = new int[10]; // Array of length 10;

int num; // User input

int count = 0; // Number of distinct numbers

// Prompt the user to enter ten numbers

System.out.print("Enter ten numbers: ");

for (int i = 0; i < 10; i++) {

num = input.nextInt();

// Test if num is distinct

if (isDistinct(distinctNumbers, num)) {

distinctNumbers[count] = num;

count++; // Increment count

}

}

// Displays the number of distinct numbers and the

// distinct numbers separated by exactly one space

System.out.println("The number of distinct numbers is " + count);

System.out.print("The distinct numbers are");

for (int i = 0; i < distinctNumbers.length; i++) {

if (distinctNumbers[i] > 0)

System.out.print(" " + distinctNumbers[i]);

}

System.out.println();

}

/\*\* Method isDistinct returns true if number is not in array false otherwise \*/

public static boolean isDistinct(int[] array, int num) {

for (int i = 0; i < array.length; i++) {

if (num == array[i])

return false;

}

return true;

}

}

\*/

public class Exercise\_10 {

static final int SIZE = 10;

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double[] numbers = new double[SIZE];

System.out.print("Enter " + SIZE + " numbers: ");

for (int i = 0; i < numbers.length; i++) numbers[i] = input.nextDouble();

System.out.println("The mean is: " + mean(numbers));

System.out.println("The standard deviation is: " + deviation(numbers));

}

public static double deviation(double[] x) {

double mean = mean(x);

double deviation = 0;

for (int i = 0; i < x.length; i++) {

deviation += Math.pow(x[i] - mean, 2);

}

return Math.sqrt(deviation / (x.length - 1));

}

public static double mean(double[] x) {

double total = 0;

for (int i = 0; i < x.length; i++) {

total += x[i];

}

System.out.println(total);

return total / x.length;

}

}

\*/

import java.util.Scanner;

public class Exercise\_05\_45 {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

double mean, // Holds the mean

deviation, // Holds the standard deviation

number; // Holds user input

mean = deviation = 0; // Set mean and deviation to 0

// Prompt the user to enter ten numbers

System.out.print("Enter ten numbers: ");

// Compute mean and standard deviation

for (int i = 1; i <= 10; i++) {

number = input.nextDouble();

mean += number;

deviation += Math.pow(number, 2);

}

deviation = Math.sqrt((deviation - (Math.pow(mean, 2) / 10)) / (10 - 1));

mean /= 10;

// Display results

System.out.println("The mean is " + mean);

System.out.printf("The standard deviation is %.5f\n", deviation);

}

}

import java.util.Scanner;

public class Exercise\_11 {

/\*\* Main method \*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter the number of students

System.out.print("Enter the number of students: ");

String[] students = new String[input.nextInt()];

int[] scores = new int[students.length];

// Prompt the user to enter each students' names and thier scores

System.out.println("Enter the name a score for each student:");

for (int i = 0; i < students.length; i++) {

System.out.print("Student " + (i + 1) + ": ");

students[i] = input.next();

System.out.print("Score: ");

scores[i] = input.nextInt();

}

// Sort student names in decreasing order of their scores

sortDecreasing(students, scores);

// Display student name in decreasing order of thier scores

for (String e: students) {

System.out.println(e);

}

}

/\*\* sorts a string and integer array in decreasing order \*/

public static void sortDecreasing(String[] strs, int[] nums) {

for (int i = 0; i < nums.length; i++) {

int max = nums[i];

int maxIndex = i;

String temp;

// Find the minimum in the list[i..nums.length - 1]

for (int j = i + 1; j < nums.length; j++) {

if (nums[j] > max) {

max = nums[j];

maxIndex = j;

}

}

if (maxIndex != i) {

// Swap string array

temp = strs[i];

strs[i] = strs[maxIndex];

strs[maxIndex] = temp;

// Swap int array

nums[maxIndex] = nums[i];

nums[i] = max;

}

}

}

}

\*/

public class Exercise\_12 {

static final int SIZE = 100000;

public static void main(String[] args) {

int[] numbers = new int[SIZE];

int key = (int)(Math.random() \* SIZE);

for (int i = 0; i < SIZE; i++) {

numbers[i] = (int)(Math.random() \* SIZE);

}

long start = System.currentTimeMillis();

int index = linearSearch(numbers, key);

long finalTime = System.currentTimeMillis() - start;

System.out.println("LinearSearch - Total time of search is: " + finalTime + " index = " + index);

start = System.currentTimeMillis();

System.out.println("Sorting array...");

sort(numbers);

System.out.println("Finished sorting.");

int index2 = binarySearch(numbers, key);

finalTime = System.currentTimeMillis() - start;

System.out.println("BINARY - Total time of search is: " + finalTime+ " index = " + index2);

}

public static int[] sort(int[] numbers) {

for (int i = 0; i < numbers.length - 1; i++) {

int swapIndex = i;

int low = numbers[i];

for (int k = i + 1; k < numbers.length; k++) {

if (low > numbers[k]) {

low = numbers[k];

swapIndex = k;

}

}

if (swapIndex != i) {

numbers[swapIndex] = numbers[i];

numbers[i] = low;

}

}

return numbers;

}

public static int linearSearch(int[] numbers, int key) {

for (int i = 0; i < numbers.length; i++) {

if (numbers[i] == key) return i;

}

return -1;

}

public static int binarySearch(int[] numbers, int key) {

int low = 0;

int high = numbers.length;

while (high >= low) {

int mid = (high + low) / 2;

if (key > numbers[mid]) {

low = mid + 1;

} else if (key == numbers[mid]) {

return mid;

} else {

high = mid - 1;

}

}

return -low - 1;

}

}

public class Exercise\_13 {

/\*\* Main method \*/

public static void main(String[] args) {

char[] board; // Create an array

// Repeat while queens are attacking

do {

// Generate a board

board = getNewBoard();

// Place eight queens

placeQueens(board);

} while (isAttacking(board));

// Display solution

print(board);

}

/\*\* placeQueens randomly places eight queens on the board\*/

public static void placeQueens(char[] board) {

int location;

for (int i = 0; i < 8; i++) {

do {

location = placeQueens();

} while (isOccupied(board[location]));

board[location] = 'Q';

}

}

/\*\* placeQueens randomly places one queen on the board \*/

public static int placeQueens() {

return (int)(Math.random() \* 64);

}

/\*\* isAttacking returns true if two queens are attacking each other \*/

public static boolean isAttacking(char[] board) {

return isSameRow(board) || isSameColumn(board) || isSameDiagonal(board);

}

/\*\* isSameRow returns true if two queens are in the same row \*/

public static boolean isSameRow(char[] board) {

int[] rows = new int[8];

for (int i = 0; i < board.length; i++) {

if (isOccupied(board[i])) {

rows[getRow(i)]++;

}

if (rows[getRow(i)] > 1)

return true;

}

return false;

}

/\*\* isSameColumn returns true if two queens are in the same column \*/

public static boolean isSameColumn(char[] board) {

int[] columns = new int[8];

for (int i = 0; i < board.length; i++) {

if (isOccupied(board[i])) {

columns[getColumn(i)]++;

}

if (columns[getColumn(i)] > 1)

return true;

}

return false;

}

/\*\* isSameDiagonal returns true if two queens are on the same diagonal \*/

public static boolean isSameDiagonal(char[] board) {

for (int i = 0; i < board.length; i++) {

if (isOccupied(board[i])) {

for (int j = 0; j < board.length; j++) {

if (isOccupied(board[j]) && Math.abs(getColumn(j) - getColumn(i)) ==

Math.abs(getRow(j) - getRow(i)) && j != i) {

return true;

}

}

}

}

return false;

}

/\*\* isOccupied returns true if the element in x is the char Q \*/

public static boolean isOccupied(char x) {

return x == 'Q';

}

/\*\* getNewBoard returns a char array filled with blank space \*/

public static char[] getNewBoard() {

char[] board = new char[64];

for (int i = 0; i < board.length; i++)

board[i] = ' ';

return board;

}

/\*\* print displays the board \*/

public static void print(char[] board) {

for (int i = 0; i < board.length; i++) {

System.out.print(

"|" + ((getRow(i + 1) == 0) ? board[i] + "|\n" : board[i]));

}

}

/\*\* getRow returns the row number that corresponds to the given index \*/

public static int getRow(int index) {

return index % 8;

}

/\*\* getColumn returns the column number that corresponds to the given index \*/

public static int getColumn(int index) {

return index / 8;

}

}

import java.util.Scanner;

public class Exercise\_14 {

/\*\* Main method \*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter a string

System.out.print("Enter a string: ");

String string = input.nextLine();

// Display the sorted string

System.out.println(sort(string));

}

/\*\* sort sorts string \*/

public static char[] sort(String s) {

char[] str = new char[s.length()]; // Create char array

// Fill array with the elements of the string

for (int i = 0; i < str.length; i++)

str[i] = s.charAt(i);

// Sort array

for (int i = 0; i < str.length - 1; i++) {

char min = str[i];

int minIndex = i;

for (int j = i + 1; j < str.length; j++) {

if (min > str[j]) {

min = str[j];

minIndex = j;

}

}

if (minIndex != i) {

str[minIndex] = str[i];

str[i] = min;

}

}

return str;

}

}

import java.util.Scanner;

public class Exercise\_15 {

/\*\* Main method \*/

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter two lists

System.out.print("Enter list1: ");

int[] list1 = new int[input.nextInt()];

for (int i = 0; i < list1.length; i++)

list1[i] = input.nextInt();

System.out.print("Enter list2: ");

int[] list2 = new int[input.nextInt()];

for (int i = 0; i < list2.length; i++)

list2[i] = input.nextInt();

// Display whether the two are strictly identical

System.out.println("Two lists are" + (equals(list1, list2) ? " " : " not ")

+ "strictly identical");

}

/\*\* equals returns true is the two arrays are indentical. False otherwise \*/

public static boolean equals(int[] list1, int[] list2) {

if (list1.length != list2.length)

return false;

for (int i = 0; i < list1.length; i++) {

if (list1[i] != list2[i])

return false;

}

return true;

}

}