//Спортсмен-лыжник начал тренировки, пробежав в первый день 10 км. Каждый  
//следующий день он увеличивал длину пробега на P процентов от пробега предыдущего  
//дня (P — вещественное, 0 < P < 50). По данному P определить, после какого дня  
//суммарный пробег лыжника за все дни превысит 200 км, и вывести найденное  
//количество дней K (целое) и суммарный пробег S (вещественное число).  
fun main(){  
 var firstDay = 10.0  
 var k = 1  
 var s = 0.0  
 while (firstDay < 200) {  
 val p: Double = *readln*().*toDouble*()  
 if (p in 1.0..49.0) {  
 firstDay += (firstDay \* p / 100)  
 k++  
 s += firstDay  
 *println*("Длина пробега: ${"%.2f".*format*(firstDay).*replace*(',','.').*toDouble*()}")  
 }  
 }  
 *println*("Кол-во дней: $k")  
 *println*("Сумарный пробег: ${"%.2f".*format*(s).*replace*(',','.').*toDouble*()}")  
}

Java:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 double firstDay = 10.0;  
 int k = 1;  
 double s = 0.0;  
 Scanner scanner = new Scanner(System.*in*);  
  
 while (firstDay < 200) {  
 double p = scanner.nextDouble();  
 if (p >= 1.0 && p <= 49.0) {  
 firstDay += (firstDay \* p / 100);  
 k++;  
 s += firstDay;  
 System.*out*.printf("Длина пробега: %.2f%n", firstDay);  
 }  
 }  
 System.*out*.println("Кол-во дней: " + k);  
 System.*out*.printf("Сумарный пробег: %.2f%n", s);  
 }  
}

//Дано вещественное число A и целое число N (> 0). Используя один цикл, вывести все  
//целые степени числа A от 1 до N.  
  
fun main(){  
 var a: Double = *readln*().*toDouble*()  
 var n: Int = *readln*().*toInt*()  
 if (n > 0)  
 {  
 for(i in 1..<n)  
 {  
 *println*("${Math.pow(a, i.toDouble())}")  
 }  
 }  
}

Java:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.in);  
 double a = Double.parseDouble(scanner.nextLine());  
 int n = Integer.parseInt(scanner.nextLine());  
  
 if (n > 0) {  
 for (int i = 1; i < n; i++) {  
 System.out.println(Math.pow(a, (double) i));  
 }  
 }  
 }  
}

//Дан набор ненулевых целых чисел; признак его завершения — число 0. Вывести  
//количество чисел в наборе.  
fun main() {  
 val arr: MutableList<String> = *mutableListOf*()  
 while (true)  
 {  
 *println*("Введите число, который вы хотите запихнуть в массив. 0 - если вы все")  
 val item = *readln*()  
 if (item != "0") {  
 arr.add(item)  
 continue  
 }  
 arr.add(item)  
 *println*("Кол-во чисел: ")  
 for (i in arr) {  
 if (i == "0") {  
 *println*(arr.indexOf(i) + 1)  
 break  
 }  
 }  
 break  
 }  
}

Java:

import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 List<String> arr = new ArrayList<>();  
 Scanner scanner = new Scanner(System.in);  
  
 while (true) {  
 System.out.println("Введите число, который вы хотите запихнуть в массив. 0 - если вы все");  
 String item = scanner.nextLine();  
 if (!item.equals("0")) {  
 arr.add(item);  
 continue;  
 }  
 arr.add(item);  
 System.out.println("Кол-во чисел: ");  
 for (String i : arr) {  
 if (i.equals("0")) {  
 System.out.println(arr.indexOf(i) + 1);  
 break;  
 }  
 }  
 break;  
 }  
 scanner.close();  
 }  
}

//Дан целочисленный массив размера N. Удалить из массива все одинаковые элементы,  
//оставив их последние вхождения.  
fun main()  
{  
 val arr: MutableList<String> = *mutableListOf*()  
 var iter = 0  
 *print*("Введите длину массива: ")  
 val n: Int = *readln*().*toInt*()  
 while (iter < n) {  
 iter += 1  
 *println*("Введите число, который вы хотите запихнуть в массив.")  
 val item = *readln*()  
 arr.add(item)  
 }  
 val lastOccurrences = *mutableListOf*<String>()  
 for (i in arr.*indices*.*reversed*()) {  
 if (!lastOccurrences.contains(arr[i])) {  
 lastOccurrences.add(arr[i])  
 }  
 }  
 lastOccurrences.*reverse*()  
 *println*("Массив без дубликатов:")  
 for (i in lastOccurrences) {  
 *println*(i)  
 }  
}

Java:

import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 List<String> arr = new ArrayList<>();  
 int iter = 0;  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Введите длину массива: ");  
 int n = Integer.*parseInt*(scanner.nextLine());  
 while (iter < n) {  
 iter += 1;  
 System.*out*.println("Введите число, который вы хотите запихнуть в массив.");  
 String item = scanner.nextLine();  
 arr.add(item);  
 }  
 List<String> lastOccurrences = new ArrayList<>();  
 for (int i = arr.size() - 1; i >= 0; i--) {  
 if (!lastOccurrences.contains(arr.get(i))) {  
 lastOccurrences.add(arr.get(i));  
 }  
 }  
 java.util.Collections.*reverse*(lastOccurrences);  
 System.*out*.println("Массив без дубликатов:");  
 for (String i : lastOccurrences) {  
 System.*out*.println(i);  
 }  
 scanner.close();  
 }  
}

//Дана матрица размера M × N и целое число K (1 ≤ K ≤ N). Удалить столбец матрицы  
//с номером K.  
fun main()  
{  
 *print*("Введите количество строк (M): ")  
 val m: Int = *readln*().*toInt*()  
 *print*("Введите количество столбцов (N): ")  
 val n: Int = *readln*().*toInt*()  
 val matrix = Array(m) **{** IntArray(n) **}** for (i in 0 *until* m) {  
 for (j in 0 *until* n) {  
 matrix[i][j] = *readln*().*toInt*()  
 }  
 }  
 *print*("Введите номер столбца K для удаления (1 ≤ K ≤ $n): ")  
 val k: Int = *readln*().*toInt*() - 1  
 for (i in 0 *until* m)  
 {  
 for (j in 0 *until* n)  
 {  
 if (j != k) {  
 *print*("${matrix[i][j]} ")  
 }  
 }  
 *println*()  
 }  
}

Java:

public class Main  
{  
 public static void main(String[] args) {  
  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.print("Введите количество строк (M): ");  
 int m = sc.nextInt();  
 System.*out*.print("Введите количество столбцов (N): ");  
 int n = sc.nextInt();  
 int[][] matrix = new int[m][n];  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 matrix[i][j] = sc.nextInt();  
 }  
 }  
 System.*out*.print("Введите номер столбца K для удаления (1 ≤ K ≤ " + n + "): ");  
 int k = sc.nextInt() - 1;  
 for (int i = 0; i < m; i++) {  
 for (int j = 0; j < n; j++) {  
 if (j != k) {  
 System.*out*.print(matrix[i][j] + " ");  
 }  
 }  
 System.*out*.println();  
 }  
 }  
}

//Даны строки S и S0. Проверить, содержится ли строка S0 в строке S. Если содержится,  
//то вывести True, если не содержится, то вывести False.  
fun main()  
{  
 val s = "qwe"  
 val s0 = "23"  
 if(s.*contains*(s0))  
 {  
 *println*("True")  
 }  
 else  
 {  
 *println*("False")  
 }  
}

Java:

public class Main {  
 public static void main(String[] args)  
 {  
 String s = "qwe";  
 String s0 = "w";  
 if (s.contains(s0)) {  
 System.out.println("True");  
 } else {  
 System.out.println("False");  
 }  
 }  
}

//Дана строка, состоящая из русских слов, разделенных пробелами (одним или  
//несколькими). Найти длину самого длинного слова.  
fun main()  
{  
 val str = "Здравствуйте Здравствуй Привет Какдела"  
 val longestWordLength = str.*split*("\\s+".*toRegex*()).*map* **{ it**.length **}**.*maxOrNull*() ?: 0  
 *println*("Длина самого длинного слова: $longestWordLength")  
}

Java:

public class Main{  
 public static void main(String[] args) {  
 String str = "Здравствуйте Здравствуй Привет Какдела";  
 int longestWordLength = Arrays.stream(str.split("\\s+")).mapToInt(String::length)  
 .max().orElse(0);  
 System.*out*.println("Длина самого длинного слова: " + longestWordLength);  
 }  
}

//Разработать функцию HowMuch(const S:string):integer, определяющую, сколько гласных  
//латинских букв в строке S стоят на нечетных местах. С помощью данной функции  
//вывести только такие строки из введенного набора, у которых по крайней мере 3  
//гласные буквы стоят на нечетных местах. Гласными считаются следующие буквы:  
//aeiouyAEIOUY  
  
fun HowMuch(S: String): Int {  
 val glass = "aeiouyAEIOUY"  
 var count = 0  
 for (i in S.*indices*) {  
 if (i % 2 == 0 && S[i] in glass)  
 {  
 count++  
 }  
 }  
 return count  
}  
fun main()  
{  
 val strings = *listOf*(  
 "Hello Worldwwqqaaaaa",  
 "Kotlin"  
 )  
  
 val filteredStrings = strings.*filter* **{** *HowMuch*(**it**) >= 3 **}** filteredStrings.*forEach* **{** *println*(**it**)  
 **}**}

Java:

import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;  
  
public class Main {  
 public static int HowMuch(String S) {  
 String glass = "aeiouyAEIOUY";  
 int count = 0;  
 for (int i = 0; i < S.length(); i++) {  
 if (i % 2 == 0 && glass.indexOf(S.charAt(i)) != -1) {  
 count++;  
 }  
 }  
 return count;  
 }  
  
 public static void main(String[] args) {  
 List<String> strings = Arrays.asList(  
 "Hello Worldwwqqaaaaa",  
 "Kotlin"  
 );  
  
 List<String> filteredStrings = new ArrayList<>();  
 for (String str : strings) {  
 if (HowMuch(str) >= 3) {  
 filteredStrings.add(str);  
 }  
 }  
  
 for (String str : filteredStrings) {  
 System.out.println(str);  
 }  
 }  
}

//Описать процедуру InvertDigits(K), меняющую порядок следования цифр целого  
//положительного числа K на обратный (K — параметр целого типа, являющийся  
//одновременно входным и выходным). С помощью этой процедуры поменять порядок  
//следования цифр на обратный для каждого из пяти данных целых чисел  
fun main() {  
 val numbers = *intArrayOf*(12345, 67890, 54321, 98765, 13579)  
 for (i in numbers.*indices*) {  
 *println*(*InvertDigits*(numbers[i]))  
 }  
}  
fun InvertDigits(K: Int): String {  
 val inverted = K.toString().*reversed*()  
 return inverted  
}

Java:

public class Main{  
 public static void main(String[] args)  
 {  
 ArrayList<Integer> InvertDigits = new ArrayList<>(Arrays.asList(12345, 67890, 54321, 98765, 13579));  
 for (int i : InvertDigits)  
 {  
 System.*out*.println(*InvertDigits*(i));  
 }  
 }  
 public static String InvertDigits(int K) {  
 return new StringBuilder(String.*valueOf*(K)).reverse().toString();  
 }  
}

open class Animal(val leave: String) {  
 open fun characteristics(): String {  
 return "Животное живет в $leave."  
 }  
}  
class Horse(leave: String, val breed: String, val name: String, val ownerName: String, val usage: String) : Animal(leave)  
{ override fun characteristics(): String {  
 return "Лошадь: порода - $breed, имя - $name, хозяин - $ownerName, использование - $usage. " +  
 super.characteristics()  
 }  
}  
class Tiger(leave: String, val rod: String, val age: Int) : Animal(leave) {  
 override fun characteristics(): String {  
 return "Тигр: род - $rod, возраст - $age лет. " +  
 super.characteristics()  
 }  
}  
class Snake(leave: String, val rod: String, val isPoisoned: Boolean) : Animal(leave) {  
 override fun characteristics(): String {  
 return "Змея: род - $rod, ядовитая - ${if (isPoisoned) "да" else "нет"}. " +  
 super.characteristics()  
 }  
}  
fun printAnimalCharacteristics(animal: Animal) {  
 *println*(animal.characteristics())  
}  
  
fun main() {  
 val horse = Horse("стабло", "Арабская", "Гроза", "Иван", "верховая езда")  
 val tiger = Tiger("лес", "Пантера", 5)  
 val snake = Snake("пустыня", "Кобра", true)  
 *printAnimalCharacteristics*(horse)  
 *printAnimalCharacteristics*(tiger)  
 *printAnimalCharacteristics*(snake)  
}

Java:

import java.util.Objects;  
  
class Animal {  
 private final String leave;  
  
 public Animal(String leave) {  
 this.leave = leave;  
 }  
  
 public String characteristics() {  
 return "Среда обитания: " + leave + ".";  
 }  
}  
  
class Horse extends Animal {  
 private final String breed;  
 private final String name;  
 private final String ownerName;  
 private final String usage;  
  
 public Horse(String leave, String breed, String name, String ownerName, String usage) {  
 super(leave);  
 this.breed = breed;  
 this.name = name;  
 this.ownerName = ownerName;  
 this.usage = usage;  
 }  
  
 @Override  
 public String characteristics() {  
 return "Лошадь: порода - " + breed + ", имя - " + name + ", хозяин - " + ownerName + ", использование - " + usage + ". " +  
 super.characteristics();  
 }  
}  
  
class Tiger extends Animal {  
 private final String rod;  
 private final int age;  
  
 public Tiger(String leave, String rod, int age) {  
 super(leave);  
 this.rod = rod;  
 this.age = age;  
 }  
  
 @Override  
 public String characteristics() {  
 return "Тигр: род - " + rod + ", возраст - " + age + " лет. " +  
 super.characteristics();  
 }  
}  
  
class Snake extends Animal {  
 private final String rod;  
 private final boolean isPoisoned;  
  
 public Snake(String leave, String rod, boolean isPoisoned) {  
 super(leave);  
 this.rod = rod;  
 this.isPoisoned = isPoisoned;  
 }  
  
 @Override  
 public String characteristics() {  
 return "Змея: род - " + rod + ", ядовитая - " + (isPoisoned ? "да" : "нет") + ". " +  
 super.characteristics();  
 }  
}  
  
public class Main {  
 public static void printAnimalCharacteristics(Animal animal) {  
 System.*out*.println(animal.characteristics());  
 }  
  
 public static void main(String[] args) {  
 Horse horse = new Horse("степь", "Арабская", "Гроза", "Иван", "верховая езда");  
 Tiger tiger = new Tiger("лес", "Пантера", 5);  
 Snake snake = new Snake("пустыня", "Кобра", true);  
 *printAnimalCharacteristics*(horse);  
 *printAnimalCharacteristics*(tiger);  
 *printAnimalCharacteristics*(snake);  
 }  
}