Задачи модуль 3

по дисциплине

“Объектно-ориентированное программирование”

Выполнила студентка

группы БФИ1901

Киселева Анна

Москва 2020

**3/6**

1. Квадратное уравнение ax2 + bx + c = 0 имеет либо 0, либо 1, либо 2 различных решения для действительных значений x. учитывая a, b и c, вы должны вернуть число решений в уравнение.

Пример:

solutions(1, 0, -1) ➞ 2

// x² - 1 = 0 has two solutions (x = 1 and x = -1).

solutions(1, 0, 0) ➞ 1

// x² = 0 has one solution (x = 0).

solutions(1, 0, 1) ➞ 0

// x² + 1 = 0 has no solutions.

import java.util.Arrays;

public class Task3 {

//1

// Квадратное уравнение ax2 + bx + c = 0 имеет либо 0,

// либо 1, либо 2 различных

//решения для действительных значений x.

// учитывая a, b и c, вы должны вернуть

//число решений в уравнение.

public static int solutions(int a, int b, int c){

int D=b\*b-4\*a\*c;

if (D>0)

return 2;

else if (D==0)

return 1;

else

return 0;

}

public static void main(String[] args) {

System.out.println("№1: " + solutions(1,0,0));

}

1. Напишите функцию, которая возвращает позицию второго вхождения " zip " в строку, или -1, если оно не происходит по крайней мере дважды. Ваш код должен быть достаточно общим, чтобы передать все возможные случаи, когда "zip" может произойти в строке.

Пример:

findZip("all zip files are zipped") ➞ 18

findZip("all zip files are compressed") ➞ -1

Примечание:

Верхний регистр " Zip "- это не то же самое, что нижний регистр "zip".

//2

// Напишите функцию, которая возвращает позицию второго вхождения " zip " в

//строку, или -1, если оно не происходит по крайней мере дважды. Ваш код должен

//быть достаточно общим, чтобы передать все возможные случаи, когда "zip" может

//произойти в строке.

public static int findZip(String strok){

int n=-1;

for (int i=0;i<strok.length()-2;i++){

if (strok.charAt(i)=='z'&& strok.charAt(i+1)=='i'&& strok.charAt(i+2)=='p'&& n==1)

n=i-1;

if (strok.charAt(i)=='z'&& strok.charAt(i+1)=='i'&& strok.charAt(i+2)=='p'&& n==-1)

n=1;

}

if (n==1)

n=-1;

return n;

}

public static void main(String[] args) {

System.out.println("№2: " + findZip("all zip files are compressed"));

}

1. Создайте функцию, которая проверяет, является ли целое число совершенным числом или нет. Совершенное число - это число, которое можно записать как сумму его множителей, исключая само число.

Например, 6-это идеальное число, так как 1 + 2 + 3 = 6, где 1, 2 и 3-Все коэффициенты 6. Точно так же 28-это совершенное число, так как 1 + 2 + 4 + 7 + 14 = 28.

Пример:

checkPerfect(6) ➞ true

checkPerfect(28) ➞ true

checkPerfect(496) ➞ true

checkPerfect(12) ➞ false

checkPerfect(97) ➞ false

//3

// Создайте функцию, которая проверяет, является ли целое число совершенным

//числом или нет. Совершенное число - это число, которое можно записать как

//сумму его множителей, исключая само число.

public static boolean checkPerfect(int a){

int sum=0;

boolean check=false;

for (int i=1;i<a-1;i++){

if (a%i==0)

sum+=i;

}

if (a==sum)

check=true;

return check;

}

public static void main(String[] args) {

System.out.println("№3: " + checkPerfect(496));

}

1. Создайте функцию, которая принимает строку и возвращает новую строку с заменой ее первого и последнего символов, за исключением трех условий:

– Если длина строки меньше двух, верните "несовместимо".".

– Если первый и последний символы совпадают, верните "два-это пара.".

Пример:

flipEndChars("Cat, dog, and mouse.") ➞ ".at, dog, and mouseC"

flipEndChars("ada") ➞ "Two's a pair."

flipEndChars("Ada") ➞ "adA"

flipEndChars("z") ➞ "Incompatible."

//4

//Создайте функцию, которая принимает строку и возвращает новую строку с

//заменой ее первого и последнего символов, за исключением трех условий:

//– Если длина строки меньше двух, верните "несовместимо".".

//– Если первый и последний символы совпадают, верните "два-это пара.".

public static String flipEndChars(String str){

char [] buf;

if(str.length()<2)

str="Two's a pair.";

else if (str.charAt(0)==str.charAt(str.length()-1)) {

str="Incompatible.";

}

else if(!((str == "Two's a pair.") || (str == "Incompatible."))) {

buf = str.toCharArray();

buf[0]=str.charAt(str.length()-1);

buf[str.length()-1]=str.charAt(0);

str=new String(buf);

}

return str;

}

public static void main(String[] args) {

System.out.println("№4: " + flipEndChars("Cat, dog, and mouse."));

}

1. Создайте функцию, которая определяет, является ли строка допустимым шестнадцатеричным кодом.

Шестнадцатеричный код должен начинаться с фунтового ключа # и иметь длину ровно 6 символов. Каждый символ должен быть цифрой от 0-9 или буквенным символом от A-F. все буквенные символы могут быть прописными или строчными.

Пример:

isValidHexCode("#CD5C5C") ➞ true

isValidHexCode("#EAECEE") ➞ true

isValidHexCode("#eaecee") ➞ true

isValidHexCode("#CD5C58C") ➞ false

// Length exceeds 6

isValidHexCode("#CD5C5Z") ➞ false

// Not all alphabetic characters in A-F

isValidHexCode("#CD5C&C") ➞ false

// Contains unacceptable character

isValidHexCode("CD5C5C") ➞ false

// Missing #

//5

// Создайте функцию, которая определяет, является ли строка допустимым

// шестнадцатеричным кодом

public static boolean isValidHexCode(String code) {

if (code.length() > 7) return false;

else if (code.matches("#[a-fA-F0-9]+")) return true;

else return false;

}

public static void main(String[] args) {

System.out.println("№5: " + isValidHexCode("#CD5C5C"));

}

1. Напишите функцию, которая возвращает true, если два массива имеют одинаковое количество уникальных элементов, и false в противном случае.

Для примера:

arr1 = [1, 3, 4, 4, 4]

arr2 = [2, 5, 7]

В arr1 число 4 появляется трижды, что означает, что оно содержит три уникальных элемента: [1, 3, 4]. Поскольку arr1 и arr2 содержат одинаковое количество уникальных элементов, этот пример вернет значение true.

Пример:

same([1, 3, 4, 4, 4], [2, 5, 7]) ➞ true

same([9, 8, 7, 6], [4, 4, 3, 1]) ➞ false

same([2], [3, 3, 3, 3, 3]) ➞ true

//6

// Напишите функцию, которая возвращает true, если два массива имеют одинаковое

// количество уникальных элементов, и false в противном случае.

public static boolean same(int [] mass, int[] mass1) {

int count = 0;

int count1 = 0;

boolean same = false;

Arrays.sort(mass);

Arrays.sort(mass1);

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

if (mass[i]!=mass[i+1])

count++;

}

for (int i=0;i<mass1.length-1;i++){

if (mass1[i]!=mass1[i+1])

count1++;

}

if (count == count1)

same = true;

return same;

}

public static void main(String[] args) {

System.out.println("№6: " + same(new int[]{1, 3, 4, 4, 4}, new int[]{2, 5, 7}));

}

1. Число Капрекара-это положительное целое число, которое после возведения в квадрат и разбиения на две лексикографические части равно сумме двух полученных новых чисел:

– Если количество цифр квадратного числа четное, то левая и правая части будут иметь одинаковую длину.

– Если количество цифр квадратного числа нечетно, то правая часть будет самой длинной половиной, а левая-самой маленькой или равной нулю, если количество цифр равно 1.

– Учитывая положительное целое число n, реализуйте функцию, которая возвращает true, если это число Капрекара, и false, если это не так.

Пример:

isKaprekar(3) ➞ false

// n² = "9"

// Left + Right = 0 + 9 = 9 ➞ 9 !== 3

isKaprekar(5) ➞ false

// n² = "25"

// Left + Right = 2 + 5 = 7 ➞ 7 !== 5

isKaprekar(297) ➞ true

// n² = "88209"

// Left + Right = 88 + 209 = 297 ➞ 297 === 297

Примечание:

Тривиально, 0 и 1-Это числа Капрекара, являющиеся единственными двумя числами, равными их квадрату.

//7

// Число Капрекара-это положительное целое число, которое после возведения в

// квадрат и разбиения на две лексикографические части равно сумме двух

// полученных новых чисел:

// – Если количество цифр квадратного числа четное, то левая и правая части будут иметь

// одинаковую длину.

// – Если количество цифр квадратного числа нечетно, то правая часть будет самой длинной

// половиной, а левая-самой маленькой или равной нулю, если количество цифр равно 1.

// – Учитывая положительное целое число n, реализуйте функцию, которая возвращает true,

// если это число Капрекара, и false, если это не так.

public static boolean isKaprekar(int n){

int c = n\*n;

String nstr=String.valueOf(n);

if (nstr.length() == 1)

return n == c;

else

return Integer.parseInt(nstr.substring(0, nstr.length()/2))+

Integer.parseInt(nstr.substring(nstr.length()/2))==n;

}

public static void main(String[] args) {

System.out.println("№7: " + isKaprekar(297));

}

1. Напишите функцию, которая возвращает самую длинную последовательность последовательных нулей в двоичной строке.

Пример:

longestZero("01100001011000") ➞ "0000"

longestZero("100100100") ➞ "00"

longestZero("11111") ➞ ""

//8

// Напишите функцию, которая возвращает самую длинную последовательность

// последовательных нулей в двоичной строке.

public static String longestZero(String n){

String sum = "";

String count = "";

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

if(n.charAt(i) == '0')

sum+="0";

else {

if (count.length()<sum.length())

count = sum;

sum = "";

}

}

if (count.length()<sum.length())count=sum;

return count;

}

public static void main(String[] args) {

System.out.println("№8: " + longestZero("01100001011000"));

}

1. Если задано целое число, создайте функцию, которая возвращает следующее простое число. Если число простое, верните само число.

Пример:

nextPrime(12) ➞ 13

nextPrime(24) ➞ 29

nextPrime(11) ➞ 11

// 11 is a prime, so we return the number itself.

//9

// Если задано целое число, создайте функцию, которая возвращает следующее

// простое число. Если число простое, верните само число.

public static int nextPrime(int num){

while(true){

boolean check=true;

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

if (num % i == 0) {

num++;

check=false;

}

}

if (check) break;

}

return num;

}

public static void main(String[] args) {

System.out.println("№9: " + nextPrime(12));

}

1. Учитывая три числа, x, y и z, определите, являются ли они ребрами прямоугольного треугольника.

Пример:

rightTriangle(3, 4, 5) ➞ true

// This is the classic example of a "nice" right angled triangle.

rightTriangle(145, 105, 100) ➞ true

// This is a less famous example.

rightTriangle(70, 130, 110) ➞ false

// This isn't a right angled triangle.

//10

// Учитывая три числа, x, y и z, определите, являются ли они ребрами

// прямоугольного треугольника.

public static boolean rigthTriangle( int x, int y, int z){

if ( x == 0 || y == 0 || z == 0 )

return false;

return ( x\*x+y\*y==z\*z || x\*x==y\*y+z\*z || x\*x+z\*z == y\*y);

}

public static void main(String[] args) {

System.out.println("№10: " + rigthTriangle(70, 130, 110));

}