Задачи модуль 4

по дисциплине

“Объектно-ориентированное программирование”

Выполнила студентка

группы БФИ1901

Киселева Анна

Москва 2020

**4/6**

1. Бесси работает над сочинением для своего класса писателей. Поскольку ее почерк довольно плох, она решает напечатать эссе с помощью текстового процессора. Эссе содержит N слов (1≤N≤100), разделенных пробелами. Каждое слово имеет длину от 1 до 15 символов включительно и состоит только из прописных или строчных букв. Согласно инструкции к заданию, эссе должно быть отформатировано очень специфическим образом: каждая строка должна содержать не более K (1≤K≤80) символов, не считая пробелов. К счастью, текстовый процессор Бесси может справиться с этим требованием, используя следующую стратегию:

– Если Бесси набирает Слово, и это слово может поместиться в текущей строке, поместите его в эту строку. В противном случае поместите слово на следующую строку и продолжайте добавлять к этой строке. Конечно, последовательные слова в одной строке все равно должны быть разделены одним пробелом. В конце любой строки не должно быть места.

* К сожалению, текстовый процессор Бесси только что сломался. Пожалуйста, помогите ей правильно оформить свое эссе!

Вам будут даны n, k и строка

Пример:

10,7 hello my name is Bessie and this is my essay➞

hello my

name is

Bessie

and this

is my

essay

import java.util.\*;

public class Task4 {

/\*1. Бесси работает над сочинением для своего класса писателей. Поскольку ее почерк довольно плох,

\* она решает напечатать эссе с помощью текстового процессора. Эссе содержит N слов (1≤N≤100),

\* разделенных пробелами. Каждое слово имеет длину от 1 до 15 символов включительно и состоит только

\* из прописных или строчных букв. Согласно инструкции к заданию, эссе должно быть отформатировано

\* очень специфическим образом: каждая строка должна содержать не более K (1≤K≤80) символов, не считая пробелов.

\* К счастью, текстовый процессор Бесси может справиться с этим требованием, используя следующую стратегию:

\* – Если Бесси набирает Слово, и это слово может поместиться в текущей строке, поместите его в эту строку.

\* В противном случае поместите слово на следующую строку и продолжайте добавлять к этой строке.

\* Конечно, последовательные слова в одной строке все равно должны быть разделены одним пробелом.

\* В конце любой строки не должно быть места.

\* – К сожалению, текстовый процессор Бесси только что сломался.

\* Пожалуйста, помогите ей правильно оформить свое эссе! \*/

public static String Bessy(int n, int k , String txt){

String[] text = txt.split(" ");

txt="";

String finaltxt="";

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

if (txt.length() + text[i].length() > k) {

finaltxt = finaltxt.trim() + "\r\n" + text[i] + " ";

txt = text[i];

} else {

finaltxt += text[i] + " ";

txt += text[i];

}

}

return finaltxt.trim();

}

public static void main(String[] args) {

System.out.println("№1 " + Bessy(3, 2, "это международный договор "));

}

1. Напишите функцию, которая группирует строку в кластер скобок. Каждый кластер должен быть сбалансирован.

Пример:

split("()()()") ➞ ["()", "()", "()"]

split("((()))") ➞ ["((()))"]

split("((()))(())()()(()())") ➞ ["((()))", "(())", "()", "()", "(()())"]

split("((())())(()(()()))") ➞ ["((())())", "(()(()()))"]

//2. Напишите функцию, которая группирует строку в кластер скобок.

//Каждый кластер должен быть сбалансирован.

public static String[] split(String str) {

List<String> list = new ArrayList<String>();

int f = 0;

int i = 0;

while (str.length() > 0) {

if (str.charAt(i) == '(') f++;

else f--;

if (f == 0) {

list.add(str.substring(0, i + 1));

str = str.substring(i + 1);

i = 0;

continue;

}

i++;

}

return list.toArray(new String[list.size()]);

}

public static void main(String[] args) {

System.out.println("№2 " + Arrays.toString(split("((()))(())()()(()())")));

}

1. Создайте две функции toCamelCase () и toSnakeCase (), каждая из которых берет одну строку и преобразует ее либо в camelCase, либо в snake\_case.

Пример:

toCamelCase("hello\_edabit") ➞ "helloEdabit"

toSnakeCase("helloEdabit") ➞ "hello\_edabit"

toCamelCase("is\_modal\_open") ➞ "isModalOpen"

toSnakeCase("getColor") ➞ "get\_color"

Примечание:

– Snake case — стиль написания составных слов, при котором несколько слов разделяются символом подчеркивания (\_), и не имеют пробелов в записи, причём каждое слово обычно пишется с маленькой буквы — «foo\_bar», «hello\_world» и т. д.

– CamelCase — стиль написания составных слов, при котором несколько слов пишутся слитно без пробелов, при этом каждое слово внутри фразы пишется с прописной буквы. Стиль получил название CamelCase, поскольку прописные буквы внутри слова напоминают горбы верблюда

//3. Создайте две функции toCamelCase () и toSnakeCase (),

// каждая из которых берет одну строку и преобразует ее либо

// в camelCase, либо в snake\_case.

public static String toCamelCase(String str) {

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

if (str.charAt(i) == '\_')

str = str.substring(0, i) + str.substring(i + 1, i + 2).toUpperCase() + str.substring(i + 2, str.length());

}

return str;

}

public static String toSnakeCase(String str) {

return str.replaceAll("([A-Z])", "\_$0").toLowerCase();

}

public static void main(String[] args) {

System.out.println("№3.1 " + toCamelCase("is\_modal\_open"));

System.out.println("№3.2 " + toSnakeCase("isModalOpen"));

}

1. Напишите функцию, которая вычисляет сверхурочную работу и оплату, связанную с сверхурочной работой.

Работа с 9 до 5: обычные часы работы

После 5 вечера это сверхурочная работа

Ваша функция получает массив с 4 значениями:

– Начало рабочего дня, в десятичном формате, (24-часовая дневная нотация)

– Конец рабочего дня. (Тот же формат)

– Почасовая ставка

– Множитель сверхурочных работ

Ваша функция должна возвращать:

$ + заработанные в тот день (округлены до ближайшей сотой)

Пример:

overTime([9, 17, 30, 1.5]) ➞ "$240.00"

overTime([16, 18, 30, 1.8]) ➞ "$84.00"

overTime([13.25, 15, 30, 1.5]) ➞ "$52.50"

Примечание:

С 16 до 17 регулярно, поэтому 1 \* 30 = 30

С 17 до 18 сверхурочно, поэтому 1 \* 30 \* 1,8 = 54

30 + 54 = 84,00 $

//4. Напишите функцию, которая вычисляет сверхурочную работу и оплату,

// связанную с сверхурочной работой.

//Работа с 9 до 5: обычные часы работы После 5 вечера это сверхурочная работа Ваша функция получает массив с 4 значениями:

//– Начало рабочего дня, в десятичном формате, (24-часовая дневная нотация)

//– Конец рабочего дня. (Тот же формат)

//– Почасовая ставка

// – Множитель сверхурочных работ

//Ваша функция должна возвращать: $ + заработанные в тот день

// (округлены до ближайшей сотой)

public static String overTime(double[] work) {

double sum = 0;

if (17 - work[0] >= 0)

sum += (17 - work[0]) \* work[2];

if (work[1] - 17 >= 0)

sum += (work[1] - 17) \* work[2] \* work[3];

return ('$' + String.valueOf(sum));

}

public static void main(String[] args) {

System.out.println("№4 " + overTime(new double[]{16, 18, 30, 1.8}));

}

1. Индекс массы тела (ИМТ) определяется путем измерения вашего веса в килограммах и деления на квадрат вашего роста в метрах. Категории ИМТ таковы:

Недостаточный вес: <18,5

Нормальный вес: 18.5-24.9

Избыточный вес: 25 и более

Создайте функцию, которая будет принимать вес и рост (в килограммах, фунтах, метрах или дюймах) и возвращать ИМТ и связанную с ним категорию. Округлите ИМТ до ближайшей десятой.

Пример:

BMI("205 pounds", "73 inches") ➞ "27.0 Overweight"

BMI("55 kilos", "1.65 meters") ➞ "20.2 Normal weight"

BMI("154 pounds", "2 meters") ➞ "17.5 Underweight"

// 5. Индекс массы тела (ИМТ) определяется путем измерения вашего веса

// в килограммах и деления на квадрат вашего роста в метрах.

// Категории ИМТ таковы:

// Недостаточный вес: <18,5

// Нормальный вес: 18.5-24.9

// Избыточный вес: 25 и более Создайте функцию, которая будет принимать вес

// и рост (в килограммах, фунтах, метрах или дюймах)

// и возвращать ИМТ и связанную с ним категорию.

// Округлите ИМТ до ближайшей десятой.

public static String BMI(String[] fat) {

double ves = Double.parseDouble(fat[0].split(" ")[0]);

double rost = Double.parseDouble(fat[1].split(" ")[0]);

String out = " ";

if (fat[0].contains("pounds"))

ves = ves \* 0.45;

if (fat[1].contains("inches"))

rost \*= 0.0254;

double BMI = Math.round((ves / (rost \* rost)) \* 10.0) / 10.0;

if (BMI < 18.5)

out = BMI + " Underweight";

if (BMI >= 18.5 && BMI <= 24.9)

out = BMI + " Normal weight";

if (BMI > 25)

out = BMI + " Overweight";

return out;

}

public static void main(String[] args) {

System.out.println("№5 " + BMI(new String[]{"55 kilos", "1.65 meters"}));

}

1. Создайте функцию, которая принимает число и возвращает его мультипликативное постоянство, которое представляет собой количество раз, которое вы должны умножать цифры в num, пока не достигнете одной цифры.

Пример:

bugger(39) ➞ 3

// Because 3 \* 9 = 27, 2 \* 7 = 14, 1 \* 4 = 4 and 4 has only one digit.

bugger(999) ➞ 4

// Because 9 \* 9 \* 9 = 729, 7 \* 2 \* 9 = 126, 1 \* 2 \* 6 = 12, and finally 1 \* 2 = 2.

bugger(4) ➞ 0

// Because 4 is already a one-digit number.

// 6. Создайте функцию, которая принимает число и возвращает его

// мультипликативное постоянство, которое представляет собой количество раз,

// которое вы должны умножать цифры в num, пока не достигнете одной цифры.

public static int bugger(int num) {

int count = 0;

while (num > 9) {

int chnum = 1;

while (num > 0) {

chnum \*= num % 10;

num /= 10;

}

num = chnum;

count++;

}

return count;

}

public static void main(String[] args) {

System.out.println("№6 " + bugger(999));

}

1. Напишите функцию, которая преобразует строку в звездную стенографию. Если символ повторяется n раз, преобразуйте его в символ\*n.

Пример:

toStarShorthand("abbccc") ➞ "ab\*2c\*3"

toStarShorthand("77777geff") ➞ "7\*5gef\*2"

toStarShorthand("abc") ➞ "abc"

toStarShorthand("") ➞ ""

// 7. Напишите функцию, которая преобразует строку в звездную стенографию.

// Если символ повторяется n раз, преобразуйте его в символ\*n.

public static String toStarShorthand(String str) {

int count = 1;

char let = str.charAt(0);

String newStr = "";

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

if (str.charAt(i) != let) {

if (count != 1)

newStr += let + "\*" + count;

else

newStr += let;

let = str.charAt(i);

count = 1;

} else

count++;

}

if (count != 1)

newStr += let + "\*" + count;

else

newStr += let;

return newStr;

}

public static void main(String[] args) {

System.out.println("№7 " + toStarShorthand("abbccc"));

}

1. Создайте функцию, которая возвращает true, если две строки рифмуются, и false в противном случае. Для целей этого упражнения две строки рифмуются, если последнее слово из каждого предложения содержит одни и те же гласные.

Пример:

doesRhyme("Sam I am!", "Green eggs and ham.") ➞ true

doesRhyme("Sam I am!", "Green eggs and HAM.") ➞ true

// Capitalization and punctuation should not matter.

doesRhyme("You are off to the races", "a splendid day.") ➞ false

doesRhyme("and frequently do?", "you gotta move.") ➞ false

Примечание:

– Без учета регистра.

– Здесь мы не обращаем внимания на такие случаи, как "thyme" и "lime".

– Мы также игнорируем такие случаи, как "away" и "today" (которые технически рифмуются, хотя и содержат разные гласные).

//8. Создайте функцию, которая возвращает true,

// если две строки рифмуются, и false в противном случае.

// Для целей этого упражнения две строки рифмуются,

// если последнее слово из каждого предложения содержит одни и те же гласные.

public static boolean doesRhyme(String str1, String str2) {

str1 = str1.substring(str1.lastIndexOf(" ") + 1);

str2 = str2.substring(str2.lastIndexOf(" ") + 1);

String let = "aeiouyAEIOUY";

String res1 = "", res2 = "";

for (int i = 0; i < str1.length(); i++) {

if (let.indexOf(str1.charAt(i)) != -1)

res1 += str1.charAt(i);

}

for (int i = 0; i < str2.length(); i++) {

if (let.indexOf(str2.charAt(i)) != -1)

res2 += str2.charAt(i);

}

if (res1.toLowerCase().equals(res2.toLowerCase())) return true;

else

return false;

}

public static void main(String[] args) {

System.out.println("№8 " + doesRhyme("Sam I am!", "Green eggs and ham."));

}

1. Создайте функцию, которая принимает два целых числа и возвращает true, если число повторяется три раза подряд в любом месте в num1 и то же самое число повторяется два раза подряд в num2.

Пример:

trouble(451999277, 41177722899) ➞ True

trouble(1222345, 12345) ➞ False

trouble(666789, 12345667) ➞ True

trouble(33789, 12345337) ➞ False

//9. Создайте функцию, которая принимает два целых числа и возвращает true,

// если число повторяется три раза подряд в любом месте в num1 и то же самое

// число повторяется два раза подряд в num2.

public static boolean trouble ( long a, long b){

String aa = Long.toString(a);

String bb = Long.toString(b);

int num = 0;

for (int i = 2 ; i < aa.length(); i++) {

if (aa.charAt(i) == aa.charAt(i -1) && aa.charAt(i) == aa.charAt(i -2))

num = aa.charAt(i);

}

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

if (bb.charAt(i)==num && bb.charAt(i+1) == num)

return true;

}

return false;

}

public static void main(String[] args) {

System.out.println("№9 " + trouble(666789, 12345667));

}

1. Предположим, что пара одинаковых символов служит концами книги для всех символов между ними. Напишите функцию, которая возвращает общее количество уникальных символов (книг, так сказать) между всеми парами концов книги.

Эта функция будет выглядеть следующим образом:

countUniqueBooks("stringSequence", "bookEnd")

Пример:

countUniqueBooks("AZYWABBCATTTA", 'A') ➞ 4

// 1st bookend group: "AZYWA" : 3 unique books: "Z", "Y", "W"

// 2nd bookend group: "ATTTA": 1 unique book: "T"

countUniqueBooks("$AA$BBCATT$C$$B$", '$') ➞ 3

countUniqueBooks("ZZABCDEF", 'Z') ➞ 0

//10. Предположим, что пара одинаковых символов служит концами книги

// для всех символов между ними. Напишите функцию,

// которая возвращает общее количество уникальных символов

// (книг, так сказать) между всеми парами концов книги.

public static int countUniqueBooks(String str, char c){

Map<Character, Integer> values = new HashMap<>();

boolean start = true;

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

if (str.charAt(i) == c && start) {

i++;

while (str.charAt(i) != c){

Integer n = values.get(str.charAt(i));

if (n == null)

values.put(str.charAt(i), 1);

else

values.put(str.charAt(i), ++n);

i++;

}

start = false;

}

if (str.charAt(i) == c)

start = true;

}

return values.size();

}

public static void main(String[] args) {

System.out.println("№10 " + countUniqueBooks("AZYWABBCATTTA", 'A'));

}