Задачи модуль 4

по дисциплине

“Объектно-ориентированное программирование”

Выполнила студентка

группы БФИ1901

Киселева Анна

Москва 2020

5/6

1. Пришло время отправлять и получать секретные сообщения.

Создайте две функции, которые принимают строку и массив и возвращают закодированное или декодированное сообщение.

Первая буква строки или первый элемент массива представляет собой символьный код этой буквы. Следующие элементы-это различия между символами: например, A +3 --> C или z -1 --> y.

Пример:

encrypt("Hello") ➞ [72, 29, 7, 0, 3]

// H = 72, the difference between the H and e is 29 (upper- and lowercase).

// The difference between the two l's is obviously 0.

decrypt([ 72, 33, -73, 84, -12, -3, 13, -13, -68 ]) ➞ "Hi there!"

encrypt("Sunshine") ➞ [83, 34, -7, 5, -11, 1, 5, -9]

import com.sun.org.apache.xerces.internal.impl.dv.util.HexBin;

import java.lang.reflect.Array;

import java.nio.charset.StandardCharsets;

import java.security.NoSuchAlgorithmException;

import java.security.MessageDigest;

import java.util.\*;

public class Task5 {

/\* 1. Пришло время отправлять и получать секретные сообщения.

Создайте две функции, которые принимают строку и массив и возвращают

закодированное или декодированное сообщение.

Первая буква строки или первый элемент массива представляет собой символьный код

этой буквы. Следующие элементы-это различия между символами: например, A +3 --> C

или z -1 --> y.\*/

public static String encrypt(String str) {

int[] mass = new int[str.length()];

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

if (i == 0)

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

else

mass[i] += str.charAt(i) - (str.charAt(i - 1));

}

return Arrays.toString(mass);

}

public static String dycrypt(int[] mass) {

char[] str = new char[mass.length];

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

if (i == 0) {

str[i] = (char) mass[i];

} else {

str[i] = (char) (mass[i - 1] + mass[i]);

mass[i] = mass[i - 1] + mass[i];

}

}

return new String(str);

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 1.1 " + encrypt("Hello"));

System.out.println("№ 1.2 " + dycrypt(new int[]{72, 33, -73, 84, -12, -3, 13, -13, -68}));

}

1. Создайте функцию, которая принимает имя шахматной фигуры, ее положение и целевую позицию. Функция должна возвращать true, если фигура может двигаться к цели, и false, если она не может этого сделать.

Возможные входные данные - "пешка", "конь", "слон", "Ладья", "Ферзь"и " король".

Пример:

canMove("Rook", "A8", "H8") ➞ true

canMove("Bishop", "A7", "G1") ➞ true

canMove("Queen", "C4", "D6") ➞ false

/\* 2. Создайте функцию, которая принимает имя шахматной фигуры, ее положение и

целевую позицию. Функция должна возвращать true, если фигура может двигаться

к цели, и false, если она не может этого сделать.\*/

public static boolean canMove(String name, String start, String end) {

char startl = start.charAt(0);

int startn = Integer.parseInt(String.valueOf(start.charAt(1)));

char endl = end.charAt(0);

int endn = Integer.parseInt(String.valueOf(end.charAt(1)));

if (startl == endl && startn == endn) return false;

switch (name) {

case "Pawn": {

if (startl == endl && startn == 2 && endn == 4)

return true;

return startl == endl && endn == (startn + 1);

}

case "Knight": {

return (Math.abs(startl - endl) == 2 && Math.abs(startn - endn) == 1) || (Math.abs(startl - endl) == 1 && Math.abs(startn - endn) == 2);

}

case "Bishop": {

return Math.abs(startl - endl) == Math.abs(startn - endn);

}

case "Rook": {

return (startl == endl && startn != endn) || (startl != endl && startn == endn);

}

case "Queen": {

if ((startl == endl && startn != endn) || (startl != endl && startn == endn))

return true;

if (Math.abs(startl - endl) == Math.abs(startn - endn))

return true;

break;

}

case "King": {

return Math.abs(startl - endl) < 2 && Math.abs(startn - endn) < 2;

}

default:

return false;

}

return false;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 2 " + canMove("Rook", "A8", "H8"));

}

1. Входная строка может быть завершена, если можно добавить дополнительные буквы, и никакие буквы не должны быть удалены, чтобы соответствовать слову. Кроме того, порядок букв во входной строке должен быть таким же, как и порядок букв в последнем слове.

Создайте функцию, которая, учитывая входную строку, определяет, может ли слово быть завершено.

Пример:

canComplete("butl", "beautiful") ➞ true

// We can add "ea" between "b" and "u", and "ifu" between "t" and "l".

canComplete("butlz", "beautiful") ➞ false

// "z" does not exist in the word beautiful.

canComplete("tulb", "beautiful") ➞ false

// Although "t", "u", "l" and "b" all exist in "beautiful", they are incorrectly ordered.

canComplete("bbutl", "beautiful") ➞ false

// Too many "b"s, beautiful has only 1.

/\* 3. Входная строка может быть завершена, если можно добавить дополнительные

буквы, и никакие буквы не должны быть удалены, чтобы соответствовать слову.

Кроме того, порядок букв во входной строке должен быть таким же, как и порядок

букв в последнем слове.\*/

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

char[] massstr1 = str1.toCharArray();

int num = 0;

for (char c : massstr1) {

if (str2.indexOf(String.valueOf(c), num) != -1)

num = str2.indexOf(String.valueOf(c), num) + 1;

else

return false;

}

return true;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 3 " + canComplite("buthl", "beautiful"));

}

1. Создайте функцию, которая принимает числа в качестве аргументов, складывает их вместе и возвращает произведение цифр до тех пор, пока ответ не станет длиной всего в 1 цифру.

Пример:

sumDigProd(16, 28) ➞ 6

sumDigProd(0) ➞ 0

sumDigProd(1, 2, 3, 4, 5, 6) ➞ 2

/\* 4. Создайте функцию, которая принимает числа в качестве аргументов, складывает их

вместе и возвращает произведение цифр до тех пор, пока ответ не станет длиной

всего в 1 цифру.\*/

public static int sumDigProd(int[] mass) {

int sum = 0;

for (int value : mass) {

sum += value;

}

while (sum > 9) {

int mult = 1;

while (sum > 0) {

mult \*= sum % 10;

sum /= 10;

}

sum = mult;

}

return sum;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

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

}

1. Напишите функцию, которая выбирает все слова, имеющие все те же гласные (в любом порядке и / или количестве), что и первое слово, включая первое слово.

Пример:

sameVowelGroup(["toe", "ocelot", "maniac"]) ➞ ["toe", "ocelot"]

sameVowelGroup(["many", "carriage", "emit", "apricot", "animal"]) ➞ ["many"]

sameVowelGroup(["hoops", "chuff", "bot", "bottom"]) ➞ ["hoops", "bot", "bottom"]

/\* 5. Напишите функцию, которая выбирает все слова, имеющие все те же гласные (в

любом порядке и / или количестве), что и первое слово, включая первое слово.\*/

public static String sameVowelGroup (String[] str) {

String vowel = "aeiouyAEIOUY";

StringBuilder first = new StringBuilder();

String second = "";

ArrayList <String> words = new ArrayList<>();

Collections.addAll(words, str);

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

if (vowel.indexOf(words.get(0).charAt(i)) != -1)

first.append(words.get(0).charAt(i));

}

for (int i = words.size() - 1; i >= 0; i--) {

for (int j = 0; j < words.get(i).length(); j++) {

if (vowel.indexOf(words.get(i).charAt(j)) != -1) {

second += words.get(i).charAt(j);

}

}

for (int k = 0; k < second.length();) {

if (first.toString().indexOf(second.charAt(k)) != -1)

k++;

else {

words.remove(i);

second = "";

}

}

}

return String.valueOf(words);

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 5 " + sameVowelGroup(new String[]{"toe", "ocelot", "maniac"}));

}

1. Создайте функцию, которая принимает число в качестве аргумента и возвращает true, если это число является действительным номером кредитной карты, а в противном случае-false.

Номера кредитных карт должны быть длиной от 14 до 19 цифр и проходить тест Луна, описанный ниже:

– Удалите последнюю цифру (это"контрольная цифра").

– Переверните число.

– Удвойте значение каждой цифры в нечетных позициях. Если удвоенное значение имеет более 1 цифры, сложите цифры вместе (например, 8 x 2 = 16 ➞ 1 + 6 = 7).

– Добавьте все цифры.

– Вычтите последнюю цифру суммы (из шага 4) из 10. Результат должен быть равен контрольной цифре из Шага 1.

Пример:

validateCard(1234567890123456) ➞ false

// Step 1: check digit = 6, num = 123456789012345

// Step 2: num reversed = 543210987654321

// Step 3: digit array after selective doubling: [1, 4, 6, 2, 2, 0, 9, 8, 5, 6, 1, 4, 6, 2, 2]

// Step 4: sum = 58

// Step 5: 10 - 8 = 2 (not equal to 6) ➞ false

validateCard(1234567890123452) ➞ true

// Same as above, but check digit checks out.

/\* 6. Создайте функцию, которая принимает число в качестве аргумента и возвращает

true, если это число является действительным номером кредитной карты, а в

противном случае-false.\*/

public static boolean validateCard(long cardNum) {

String str="";

if ( Long.toString(cardNum).length()>= 14 && Long.toString(cardNum).length() <= 19) {

// step 1

long lastNum = cardNum%10;

StringBuffer cardNumStr = new StringBuffer(Long.toString(cardNum/=10));

// step 2

cardNumStr.reverse();

// step 3

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

if (i%2==0){

int c =Character.getNumericValue(cardNumStr.charAt(i))\*2;

if(c>9){

String buf = Integer.toString(c);

str += Character.getNumericValue(buf.charAt(0))+Character.getNumericValue(buf.charAt(1));

}

else str+=c;

}

else str+=cardNumStr.charAt(i);

}

System.out.println(str);

int sum=0;

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

sum+=Character.getNumericValue(str.charAt(i));

System.out.println(sum);

System.out.println(lastNum);

if (lastNum==10-sum%10)

return true;

}

return false;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 6 " + validateCard(1234567890123452L));

}

1. Напишите функцию, которая принимает положительное целое число от 0 до 999 включительно и возвращает строковое представление этого целого числа, написанное на английском языке.

Пример:

numToEng(0) ➞ "zero"

numToEng(18) ➞ "eighteen"

numToEng(126) ➞ "one hundred twenty six"

numToEng(909) ➞ "nine hundred nine"

Тоже самое нужно сделать и для русского языка.

/\* 7. Напишите функцию, которая принимает положительное целое число от 0 до 999

включительно и возвращает строковое представление этого целого числа,

написанное на английском языке.\*/

public static String numToEng(int num) {

String strnum = "";

if (num == 0) return "zero";

// сотни

switch (num / 100) {

case 1: {

strnum += "one hundred ";

break;

}

case 2: {

strnum += "two hundred ";

break;

}

case 3: {

strnum += "three hundred ";

break;

}

case 4: {

strnum += "four hundred ";

break;

}

case 5: {

strnum += "five hundred ";

break;

}

case 6: {

strnum += "six hundred ";

break;

}

case 7: {

strnum += "seven hundred ";

break;

}

case 8: {

strnum += "eight hundred ";

break;

}

case 9: {

strnum += "nine hundred ";

break;

}

}

// десятки до 20

switch (num / 10 % 10) {

case 1: {

switch (num % 10) {

case 0: {

strnum += "ten";

return strnum;

}

case 1: {

strnum += "eleven";

return strnum;

}

case 2: {

strnum += "twelve";

return strnum;

}

case 3: {

strnum += "thirteen";

return strnum;

}

case 4: {

strnum += "fourteen";

return strnum;

}

case 5: {

strnum += "fifteen";

return strnum;

}

case 6: {

strnum += "sixteen";

return strnum;

}

case 7: {

strnum += "seventeen";

return strnum;

}

case 8: {

strnum += "eighteen";

return strnum;

}

case 9: {

strnum += "nineteen";

return strnum;

}

}

}

// десятки после 20

case 2: {

strnum += "twenty ";

break;

}

case 3: {

strnum += "thirty ";

break;

}

case 4: {

strnum += "forty ";

break;

}

case 5: {

strnum += "fifty ";

break;

}

case 6: {

strnum += "sixty ";

break;

}

case 7: {

strnum += "seventy ";

break;

}

case 8: {

strnum += "eighty ";

break;

}

case 9: {

strnum += "ninety ";

break;

}

}

// единицы

switch (num % 10) {

case 1: {

strnum += "one";

break;

}

case 2: {

strnum += "two";

break;

}

case 3: {

strnum += "three";

break;

}

case 4: {

strnum += "four";

break;

}

case 5: {

strnum += "five";

break;

}

case 6: {

strnum += "six";

break;

}

case 7: {

strnum += "seven";

break;

}

case 8: {

strnum += "eight";

break;

}

case 9: {

strnum += "nine";

break;

}

}

return strnum;

}

public static String numToRus(int num) {

String strnum = "";

if (num == 0) return "ноль";

// сотни

switch (num / 100) {

case 1: {

strnum += "сто ";

break;

}

case 2: {

strnum += "двести ";

break;

}

case 3: {

strnum += "триста ";

break;

}

case 4: {

strnum += "четыреста ";

break;

}

case 5: {

strnum += "пятьсот ";

break;

}

case 6: {

strnum += "шестьсот ";

break;

}

case 7: {

strnum += "семьсот ";

break;

}

case 8: {

strnum += "восемьсот ";

break;

}

case 9: {

strnum += "девятьсот ";

break;

}

}

// десятки до 20

switch (num / 10 % 10) {

case 1: {

switch (num % 10) {

case 0: {

strnum += "десять";

return strnum;

}

case 1: {

strnum += "одиннадцать";

return strnum;

}

case 2: {

strnum += "двенадцать";

return strnum;

}

case 3: {

strnum += "тринадцать";

return strnum;

}

case 4: {

strnum += "четырнадцать";

return strnum;

}

case 5: {

strnum += "пятнадцать";

return strnum;

}

case 6: {

strnum += "шестнадцать";

return strnum;

}

case 7: {

strnum += "семнадцать";

return strnum;

}

case 8: {

strnum += "восемьнадцать";

return strnum;

}

case 9: {

strnum += "двадцать";

return strnum;

}

}

}

// десятки после 20

case 2: {

strnum += "двадцать ";

break;

}

case 3: {

strnum += "тридцать ";

break;

}

case 4: {

strnum += "сорок ";

break;

}

case 5: {

strnum += "пятьдесят ";

break;

}

case 6: {

strnum += "шестьдесят ";

break;

}

case 7: {

strnum += "семьдесят ";

break;

}

case 8: {

strnum += "восемьдесят ";

break;

}

case 9: {

strnum += "девяносто ";

break;

}

}

// единицы

switch (num % 10) {

case 1: {

strnum += "один";

break;

}

case 2: {

strnum += "два";

break;

}

case 3: {

strnum += "три";

break;

}

case 4: {

strnum += "четыре";

break;

}

case 5: {

strnum += "пять";

break;

}

case 6: {

strnum += "шесть";

break;

}

case 7: {

strnum += "семь";

break;

}

case 8: {

strnum += "восемь";

break;

}

case 9: {

strnum += "девять";

break;

}

}

return strnum;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 7.1 " + numToEng(108));

System.out.println("№ 7.2 " + numToRus(108));

}

1. Хеш-алгоритмы легко сделать одним способом, но по существу невозможно сделать наоборот. Например, если вы хешируете что-то простое, например, password123, это даст вам длинный код, уникальный для этого слова или фразы. В идеале, нет способа сделать это в обратном порядке. Вы не можете взять хеш-код и вернуться к слову или фразе, с которых вы начали.

Создайте функцию, которая возвращает безопасный хеш SHA-256 для данной строки. Хеш должен быть отформатирован в виде шестнадцатеричной цифры.

Пример:

getSha256Hash("password123") ➞ "ef92b778bafe771e89245b89ecbc08a44a4e166c06659911881f383d4473e94f"

getSha256Hash("Fluffy@home") ➞ "dcc1ac3a7148a2d9f47b7dbe3d733040c335b2a3d8adc7984e0c483c5b2c1665"

getSha256Hash("Hey dude!") ➞ "14f997f08b8ad032dcb274198684f995d34043f9da00acd904dc72836359ae0f"

Примечание:

Бонус, если вы можете сделать это без импорта каких-либо библиотек ;)

/\*8. Хеш-алгоритмы легко сделать одним способом, но по существу невозможно

сделать наоборот. Например, если вы хешируете что-то простое, например,

password123, это даст вам длинный код, уникальный для этого слова или фразы. В

идеале, нет способа сделать это в обратном порядке. Вы не можете взять хеш-код и

вернуться к слову или фразе, с которых вы начали.\*/

public static String getSha256Hash(String str)throws NoSuchAlgorithmException {

MessageDigest digest = MessageDigest.getInstance("SHA-256");

byte[] text=digest.digest(str.getBytes(StandardCharsets.UTF\_8));

return HexBin.encode(text).toLowerCase();

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 8 " + getSha256Hash("password123"));

}

1. Напишите функцию, которая принимает строку и возвращает строку с правильным регистром для заголовков символов в серии "Игра престолов".

Слова and, the, of и in должны быть строчными. Все остальные слова должны иметь первый символ в верхнем регистре, а остальные-в Нижнем.

Пример:

correctTitle("jOn SnoW, kINg IN thE noRth.")

➞ "Jon Snow, King in the North."

correctTitle("sansa stark, lady of winterfell.")

➞ "Sansa Stark, Lady of Winterfell."

correctTitle("TYRION LANNISTER, HAND OF THE QUEEN.")

➞ "Tyrion Lannister, Hand of the Queen."

Примечания:

– Знаки препинания и пробелы должны оставаться в своих первоначальных положениях.

– Дефисные слова считаются отдельными словами.

– Будьте осторожны со словами, которые содержат and, the, of или in.

/\* 9. Напишите функцию, которая принимает строку и возвращает строку с правильным

регистром для заголовков символов в серии "Игра престолов".

Слова and, the, of и in должны быть строчными. Все остальные слова должны иметь

первый символ в верхнем регистре, а остальные-в Нижнем.\*/

public static String correctTitle(String title) {

String[] text = title.toLowerCase().split(" ");

title = "";

String pos = "";

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

while (text[i].contains("-")) {

pos += text[i].indexOf("-") + " ";

text[i] = text[i].substring(0, text[i].indexOf("-")) + " " + text[i].substring(text[i].indexOf("-") + 1);

System.out.println(text[i].substring(text[i].indexOf("-") + 1));

}

if (text[i].equals("in") || text[i].equals("of") || text[i].equals("and") || text[i].equals("the"))

title += text[i].toLowerCase() + " ";

else

title += text[i].substring(0, 1).toUpperCase() + text[i].substring(1) + " ";

}

return title;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 9 " + correctTitle("TYRION LANNISTER, HAND OF THE QUEEN."));

}

1. Как указано в онлайн-энциклопедии целочисленных последовательностей:

Гексагональная решетка - это привычная двумерная решетка, в которой каждая точка имеет 6 соседей.

Центрированное шестиугольное число - это центрированное фигурное число, представляющее шестиугольник с точкой в центре и всеми другими точками, окружающими центральную точку в шестиугольной решетке.

Illustration of initial terms:

.

. o o o o

. o o o o o o o o

. o o o o o o o o o o o o

. o o o o o o o o o o o o o o o o

. o o o o o o o o o o o o

. o o o o o o o o

. o o o o

.

. 1 7 19 37

.

Напишите функцию, которая принимает целое число n и возвращает "недопустимое", если n не является центрированным шестиугольным числом или его иллюстрацией в виде многострочной прямоугольной строки в противном случае.

Пример:

hexLattice(1) ➞ " o "

// o

hexLattice(7) ➞ " o o \n o o o \n o o "

// o o

// o o o

// o o

hexLattice(19) ➞ " o o o \n o o o o \n o o o o o \n o o o o \n o o o "

// o o o

// o o o o

// o o o o o

// o o o o

// o o o

hexLattice(21) ➞ "Invalid"

/\* 10. Как указано в онлайн-энциклопедии целочисленных последовательностей:

Гексагональная решетка - это привычная двумерная решетка, в которой каждая точка

имеет 6 соседей.

Центрированное шестиугольное число - это центрированное фигурное число,

представляющее шестиугольник с точкой в центре и всеми другими точками,

окружающими центральную точку в шестиугольной решетке. \*/

public static String haxLattice(int n){

int num = 1;

int i = 1;

String res="";

String str2="";

while (n>num) {

i++;

num = 3 \* i \* (i - 1) + 1;

}

int l = i;

// верхняя половина

if (n != num)

res = "invalid";

else {

while (l < i \* 2 - 1) {

for (int a = 0; a < i \* 2 - 1 - l; a++)

res += " ";

for (int b = 0; b < l; b++)

res += " o ";

res += "\n";

l++;

}

// нижняя половина

while (l >= i) {

for (int a = 0; a < i \* 2 - 1 - l; a++)

res += " ";

for (int b = l; b > 0; b--)

res += " o ";

res += "\n";

l--;

}

}

return res;

}

public static void main(String[] args) throws NoSuchAlgorithmException {

System.out.println("№ 10 \n" + haxLattice(19));

}