**Task 6**

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

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.

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.

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

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

Пример:

split("()()()") ➞ ["()", "()", "()"]

split("((()))") ➞ ["((()))"]

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

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

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("") ➞ ""

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).

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]

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"]

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.

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"

1. В «поросячей латыни» (свинский латинский) есть два очень простых правила:

– Если слово начинается с согласного, переместите первую букву (буквы) слова до гласного до конца слова и добавьте «ay» в конец.

have ➞ avehay

cram ➞ amcray

take ➞ aketay

cat ➞ atcay

shrimp ➞ impshray

trebuchet ➞ ebuchettray

* Если слово начинается с гласной, добавьте "yay" в конце слова.

ate ➞ ateyay

apple ➞ appleyay

oaken ➞ oakenyay

eagle ➞ eagleyay

Напишите две функции, чтобы сделать переводчик с английского на свинский латинский. Первая функция translateWord (word) получает слово на английском и возвращает это слово, переведенное на латинский язык. Вторая функция translateSentence (предложение) берет английское предложение и возвращает это предложение, переведенное на латинский язык.

Пример:

translateWord("flag") ➞ "agflay"

translateWord("Apple") ➞ "Appleyay"

translateWord("button") ➞ "uttonbay"

translateWord("") ➞ ""

translateSentence("I like to eat honey waffles.") ➞ "Iyay ikelay otay eatyay oneyhay afflesway."

translateSentence("Do you think it is going to rain today?") ➞ "Oday youyay inkthay ityay isyay oinggay otay ainray odaytay?"

Примечание:

– Регулярные выражения помогут вам не исказить пунктуацию в предложении.

– Если исходное слово или предложение начинается с заглавной буквы, перевод должен сохранить свой регистр

1. Напишите функцию, которая возвращает самую длинную неповторяющуюся подстроку для строкового ввода.

Пример:

longestNonrepeatingSubstring("abcabcbb") ➞ "abc"

longestNonrepeatingSubstring("aaaaaa") ➞ "a"

longestNonrepeatingSubstring("abcde") ➞ "abcde"

longestNonrepeatingSubstring("abcda") ➞ "abcd"

Примечание:

– Если несколько подстрок связаны по длине, верните ту, которая возникает первой.

– Бонус: можете ли вы решить эту проблему в линейном времени?

1. Создайте функцию, которая принимает арабское число и преобразует его в римское число.

Пример:

convertToRoman(2) ➞ "II"

convertToRoman(12) ➞ "XII"

convertToRoman(16) ➞ "XVI"

Примечание:

– Все римские цифры должны быть возвращены в верхнем регистре.

– Самое большое число, которое может быть представлено в этой нотации, - 3,999.

1. Создайте функцию, которая принимает строку и возвращает true или false в зависимости от того, является ли формула правильной или нет.

Пример:

formula("6 \* 4 = 24") ➞ true

formula("18 / 17 = 2") ➞ false

formula("16 \* 10 = 160 = 14 + 120") ➞ false