1/6

1. Создайте функцию, которая принимает целое число галлонов и преобразует его в литры.

Пример:

```
convert(5) → 18.925

convert(3) → 11.355

convert(8) → 30.28
```

2. Вы пишете программу для квази-фитнес-приложения и хотите создать функцию для расчета калорий, сожженных пользователем во время тренировки. Функция должна принимать время тренировки в минутах и интенсивность, где 1 – низкая интенсивность, 2 – средняя, 3 – высокая, а затем вычислять количество сожженных калорий на основе этой информации.

Пример:

```
fitCalc(15, 1) → 15

fitCalc(24, 2) → 48

fitCalc(41, 3) → 123
```

3. В этой задаче вы управляете складом, где хранятся товары трех типов:

- Коробки содержат по 20 товаров в каждой.
- Мешки содержат по 50 товаров в каждом.
- Бочки содержат по 100 товаров в каждой.

Вам предоставили информацию о количестве каждого типа емкостей на складе, и вам нужно создать функцию, которая вернет общее количество товаров на складе, учитывая объекты хранения разных типов.

Пример:

```
containers(3, 4, 2) → 460

containers(5, 0, 2) → 300

containers(4, 1, 4) → 530
```

4. Создайте функцию, которая принимает 3 числа: X, Y и Z. Эти числа представляют длины сторон треугольника. Функция должна вернуть тип треугольника на основе данных сторон: "равносторонний" (если все стороны равны), "равнобедренный" (если две стороны равны), "разносторонний" (если все стороны разные) или "не является треугольником" (если невозможно построить треугольник с заданными сторонами).

Пример:

```
triangleType(5, 5, 5) → isosceles

triangleType(5, 4, 5) → equilateral
```

```
triangleType(3, 4, 5) → different-sided

triangleType(5, 1, 1) → not a triangle
```

5. В Java есть вариация условного оператора – тернарный оператор "? :", принимающий три операнда и возвращающий один из них на основе значения условия. Он имеет следующую структуру:

условие ? выражение1 : выражение2

Ваша задача создать функцию, которая принимает два числа a и b, а затем с помощью тернарного оператора определяет, какое из чисел больше, и возвращает большее число.

Пример:

```
ternaryEvaluation(8, 4) → 8

ternaryEvaluation(1, 11) → 11

ternaryEvaluation(5, 9) → 9
```

6. У меня есть ограниченное количество ткани определенной длины, и я хочу сшить как можно больше пододеяльников. Создайте функцию, которая будет принимать длину ткани (в метрах) и размер одной детали (ширина и длина в метрах), а затем возвращать количество пододеяльников, которые я смогу сшить, прежде чем кончится рулон.

n * 2 – это количество квадратных метров имеющейся ткани,
w и h – это длина и ширина одной детали в метрах

Пример:

```
howManyItems(22, 1.4, 2) → 3

howManyItems(45, 1.8, 1.9) → 6

howManyItems(100, 2, 2) → 12
```

Примечание:
- Не считайте пододеяльник, если на него не хватило ткани в рулоне

7. Напишите функцию, вычисляющую факториал выбранного числа.

Пример:

```
factorial(3) → 6

factorial(5) → 120

factorial(7) → 5040
```

8. Создайте функцию, которая находит наибольший общий делитель двух чисел.

Пример:

```
gcd(48, 18) → 6

gcd(52, 8) → 4

gcd(259, 28) → 1
```

9. Создайте функцию, которая принимает количество билетов на концерт, проданных через веб-сервис, и стоимость одного билета с учетом фиксированной комиссии. Функция должна вернуть общую выручку от продажи билетов.

Пример:

```
ticketSaler(70, 1500) → 75600

ticketSaler(24, 950) → 16416

ticketSaler(53, 1250) → 47700
```

10. Создайте функцию, которая принимает целое число студентов и количество парт в аудитории. Функция должна определить, сколько столов не хватает для размещения всех студентов, если за одним столом помещается два студента.

Пример:

```
tables(5, 2) → 1

tables(31, 20) → 0

tables(123, 58) → 4
```

2/6

1. Создайте функцию, которая определяет, есть ли в строке повторяющиеся символы.

Пример:

```
duplicateChars("Donald") → true

duplicateChars("orange") → false
```

2. Создайте метод, который принимает строку (фамилию и имя человека) и возвращает строку с инициалами без пробелов.

Пример:

```
getInitials("Ryan Gosling") → "RG"

getInitials("Barack Obama") → "BA"
```

3. Создайте функцию, которая принимает массив и возвращает разницу между суммой четных и нечетных.

Пример:

```
differenceEvenOdd([44, 32, 86, 19]) → 143

differenceEvenOdd([22, 50, 16, 63, 31, 55]) → 61
```

4. Создайте функцию, которая принимает массив и возвращает true, если в массиве есть хотя бы один элемент, который равен среднему арифметическому всех элементов массива, и false в противном случае.

Пример:

```
equalToAvg([1, 2, 3, 4, 5]) → true

equalToAvg ([1, 2, 3, 4, 6]) → false
```

5. Создайте метод, который берет массив целых чисел и возвращает массив, в котором каждое целое число умножено на индекс этого числа в массиве.

Пример:

```
indexMult([1, 2, 3]) → [0, 2, 6]

indexMult([3, 3, -2, 408, 3, 31]) → [0, 3, -4, 1224, 12, 155]
```

6. Создайте метод, который принимает строку в качестве аргумента и возвращает ее в обратном порядке.

Пример:

```
reverse("Hello World") → "dlroW olleH"

reverse("The quick brown fox.") → ".xof nworb kciuq ehT"
```

7. Создайте функцию, которая при заданном числе возвращает соответствующее число Трибоначчи. Последовательность Трибоначчи начинается с элементов «0, 0, 1».

Пример:

```
Tribonacci(7) → 7

Tribonacci(11) → 81
```

8. Хэш-суммы в системе контроля версий (например, Git) выглядят как уникальная строка из символов (от *a* до *f*) и цифр (от 0 до 9) длиной в 40 элементов. В Git используется SHA-1 хэш-функция для создания хэшей коммитов.

   Создайте функцию, генерирующую квази-хэш заданной пользователем длины.

Пример:

```
pseudoHash(5) → "04bf2"

pseudoHash(10) → "2d9c45e1f3"

pseudoHash(0) → ""
```

9. Напишите функцию, которая находит слово "help" в данной строке-стенограмме автоматизированного телефонного диспетчера службы спасения. Ответьте "Вызов сотрудника", если слово найдено, в противном случае – "Продолжайте ожидание".

Пример:

```
botHelper("Hello, I'm under the water, please help me") → "Calling for a
staff member"

botHelper("Two pepperoni pizzas please") → "Keep waiting"
```

Примечание:
Строка "help" может появляться в разных случаях символов (например, в верхнем, нижнем регистре).

10. Создайте функцию, которая принимает две строки и определяет, являются ли они анаграммами.

Пример:

```
isAnagram("listen", "silent") → true

isAnagram("eleven plus two", "twelve plus one") → true

isAnagram("hello", "world") → false
```

3/6

1. Создайте функцию, которая принимает строку и заменяет все гласные буквы на символ «*».

Пример:

```
replaceVovels("apple") → "*ppl*"

duplicateChars("Even if you did this task not by yourself, you have to understand every single line of code.") → "*v*n *f *** d*d th*s t*sk n*t b* ***rs*lf, *** h*v* t* *nd*rst*nd *v*r* s*ngl* l*n* *f c*d*."
```

2. Напишите функцию, которая принимает строку и заменяет две идущие подряд буквы по шаблону «Double*».

Пример:

```
stringTransform("hello") → "heDoubleLo"

stringTransform("bookkeeper") → "bDoubleODoubleKDoubleEper"
```

3. Помогите ребенку разобраться с игрушкой на развитие - поместится ли параллелепипед в коробку с отверстиями определенных параметров. Напишите функцию, которая принимает три измерения игрушечного блока: высоту(a), ширину(b) и глубину(c) и возвращает true, если этот блок может поместиться в отверстие с шириной(w) и высотой(h).

Пример:

```
doesBlockkFit(1, 3, 5, 4, 5) → true

doesBlockFit(1, 8, 1, 1, 1) → true

doesBlockFit(1, 2, 2, 1, 1) → false
```

Примечание:
- Вы можете повернуть блок любой стороной к отверстию.
- Мы предполагаем, что блок подходит, если его размеры равны размерам отверстия, а не строго меньше.
- Блок можно класть только под прямым углом к поверхности.

4. Создайте функцию, которая принимает число в качестве входных данных и возвращает true, если сумма квадратов его цифр имеет ту же четность, что и само число. В противном случае верните false.

Пример:

```
numCheck(243) → true
// 243 нечетное, как и 29 (2^2 = 4, 4^2 = 16, 3^2 = 9, 4+16+9 = 29)

numCheck(52) → false
// 52 четное, но 29 – нет (5^2=25, 2^2=4, 25 + 4 = 29)
```

5. Создайте метод, который берет массив целых чисел-коэффициентов и возвращает количество целочисленных корней квадратного уравнения.

Пример:

```
countRoots([1, -3, 2]) → 2

countRoots([2, 5, 2]) → 1

countRoots([1, -6, 9]) → 1
```

6. Создайте метод, который принимает двумерный массив, представляющий информацию о продажах разных товаров в различных магазинах, и возвращает товары, которые были проданы в каждом из магазинов.

Пример:

```
salesData([
["Apple", "Shop1", "Shop2", "Shop3", "Shop4"],
["Banana", "Shop2", "Shop3", "Shop4"],
["Orange", "Shop1", "Shop3", "Shop4"],
["Pear", "Shop2", "Shop4"]
]) → ["Apple"]

salesData([
["Fridge", "Shop2", "Shop3"],
["Microwave", "Shop1", "Shop2", "Shop3", "Shop4"],
["Laptop", "Shop3", "Shop4"],
["Phone", "Shop1", "Shop2", "Shop3", "Shop4"]
]) → ["Microwave", "Phone"]
```

7. Создайте функцию, которая определяет, можно ли разбить заданное предложение на слова так, чтобы каждое слово начиналось с последней буквы предыдущего слова.

Пример:

```
validSplit("apple eagle egg goat") → true

validSplit("cat dog goose fish") → false
```

8. Напишите метод, который определяет, является ли заданный массив «волнообразным». Последовательность чисел считается волнообразной, если разница между соседними элементами чередуется между убыванием и возрастанием.

Пример:

```
waveForm([3, 1, 4, 2, 7, 5]) → true
последовательность начинается с убывания (3, 1), сменяющегося на
возрастание (1, 4) и т.д.

waveForm([1, 2, 3, 4, 5]) → false

waveForm([1, 2, -6, 10, 3]) → true
```

9. Напишите функцию, которая находит наиболее часто встречающуюся гласную в предложении.

Пример:

```
commonVovel("Hello world") ➞ "o"

commonVovel("Actions speak louder than words.") ➞ "a"
```

10. Создайте функцию, которая принимает n целочисленных массивов длины n, а затем изменяет каждый n-ый элемент n-го массива на среднее арифметическое элементов n-го столбца остальных массивов.

Пример:

```
dataScience([
[1, 2, 3, 4, 5],
[6, 7, 8, 9, 10],
[5, 5, 5, 5, 5],
[7, 4, 3, 14, 2],
[1, 0, 11, 10, 1]
]) ➞
[[5, 2, 3, 4, 5],
[6, 3, 8, 29, 10],
[5, 5, 6, 5, 35],
[7, 4, 3, 12, 2],
[1, 0, 11, 10, 13]]

dataScience([
[6, 4, 19, 0, 0],
[81, 25, 3, 1, 17],
[48, 12, 60, 32, 14],
[91, 47, 16, 65, 217],
[5, 73, 0, 4, 21]
]) ➞
[[56, 4, 19, 0, 0],
[81, 34, 3, 1, 17],
[48, 12, 10, 32, 14],
[91, 47, 16, 9, 217],
[5, 73, 0, 4, 62]]
```

4/6

1. Напишите рекурсивную функцию, которая принимает строку и удаляет из неё повторяющиеся символы. Функция должна вернуть строку, в которой каждый символ встречается только один раз.

Пример:

```
nonRepeatable("abracadabra") → " abrcd"

nonRepeatable("paparazzi") → " parzi"
```

2. Напишите функцию, которая генерирует все возможные правильные комбинации пар скобок для заданного числа n.

Пример:

```
generateBrackets(1) → ["()"]

generateBrackets(2) → ["(())", "()()"]

generateBrackets(3) → ["((()))", "(()())", "(())()", "()(())", "()()()"]
```

3. Напишите функцию, которая генерирует все возможные бинарные комбинации длины n, в которых не может быть соседствующих нулей.

Пример:

```
binarySystem(3) → ["010", "011", "101", "110", "111"]

binarySystem(4) → ["0101", "0110", "0111", "1010", "1011", "1101",
"1110", "1111"]
```

4. Реализуйте функцию, которая принимает строку и возвращает длину самого длинного последовательного ряда в этом массиве. Последовательный ряд – это список соседних элементов, идущих подряд в алфавитном порядке, который может быть как увеличивающимся, так и уменьшающимся.

Пример:

```
alphabeticRow("abcdjuwx") → "abcd"
// два последовательных ряда: "abcd", "uwx"; самый длинный: "abcd"

alphabeticRow("klmabzyxw") → "zyxw"
```

5. Напишите функцию, которая принимает строку и подсчитывает количество идущих подряд символов, заменяя соответствующим числом повторяющиеся символы. Отсортируйте строку по возрастанию длины буквенного паттерна.

Пример:

```
("aaabbcdd") → "c1b2d2a3"

("vvvvaajaaaaa") → "j1a2v4a5"
```

6. Напишите функцию, принимающую положительное целое число в строковом формате, не превышающее 1000, и возвращающую его целочисленное представление.

Пример:

```
convertToNum("eight") → 8

convertToNum("five hundred sixty seven") → 567
```

```
convertToNum("thirty one") → 31
```

7. Напишите функцию, принимающую строку цифр, выполняющую поиск подстроки максимальной длины с уникальными элементами. Если найдено несколько подстрок одинаковой длины, верните первую.

Пример:

```
uniqueSubstring("123412324") → "1234"

uniqueSubstring("111111") → "1"

uniqueSubstring("77897898") → "789"
```

8. Напишите функцию поисковик наименьшего матричного пути. На вход поступает двумерный массив, размера n x n, ваша задача найти путь с минимальной суммой чисел, передвигаясь только вправо или вниз.

Пример:

```
shortestWay(
[[1, 3, 1],
 [1, 5, 1],
 [4, 2, 1]]) → 7
// 1+3+1+1+1=7

shortestWay(
[[2, 7, 3],
 [1, 4, 8],
 [4, 5, 9]]) → 21
```

9. Создайте функцию, принимающую строку, содержащую числа внутри слов. Эти числа представляют расположение слова для новой строящейся строки.

Пример:

```
numericOrder("t3o the5m 1One all6 r4ule ri2ng") → " One ring to rule them
all"

numericOrder("re6sponsibility Wit1h gr5eat power3 4comes g2reat") → " With
great power comes great responsibility"
```

10. Напишите функцию, принимающую два числа, которая делает второе число максимально возможным за счет замены своих элементов элементами первого числа. Брать цифру можно только один раз.

Пример:

```
switchNums(519, 723) → 953

switchNums(491, 3912) → 9942

switchNums(6274, 71259) → 77659
```

5/6

1. Создайте функцию, которая возвращает true, если две строки имеют один и тот же буквенный шаблон, и false в противном случае.

Пример:

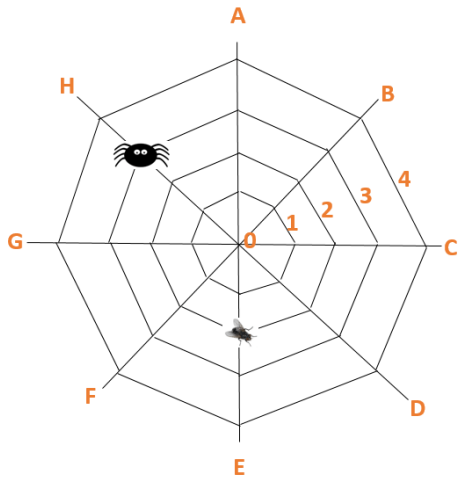
```
sameLetterPattern("ABAB", "CDCD") → true

sameLetterPattern("ABCBA", "BCDCB") → true

sameLetterPattern("FFGG", "CDCD") → false

sameLetterPattern("FFFF", "ABCD") → false
```

2. Паутина определяется кольцами, пронумерованными от 0 до 4 от центра, и радиалами, помеченными по часовой стрелке сверху как A-H.

Создайте функцию, которая принимает координаты паука и мухи и возвращает кратчайший путь для паука, чтобы добраться до мухи.

Стоит отметить, что кратчайший путь должен быть рассчитан "геометрически", а не путем подсчета количества точек, через которые проходит этот путь.

- Угол между каждой парой радиалов одинаков (45 градусов).
- Расстояние между каждой парой колец всегда одинаково (скажем, "x").



На приведенном выше рисунке координаты паука - "H3", а координаты мухи - "E2". Паук будет следовать по кратчайшему пути "H3-H2-H1-A0-E1-E2", чтобы добраться до мухи.

Пример:

```
spiderVsFly("H3", "E2") → "H3-H2-H1-A0-E1-E2"

spiderVsFly("A4", "B2") → "A4-A3-A2-B2"

spiderVsFly("A4", "C2") → "A4-A3-A2-B2-C2"
```

3. Создайте функцию, которая будет рекурсивно подсчитывать количество цифр числа. Преобразование числа в строку не допускается, поэтому подход является рекурсивным.

```
digitsCount(4666) → 4
```

```
digitsCount(544) → 3

digitsCount(121317) → 6

digitsCount(0) → 1

digitsCount(12345) → 5

digitsCount(1289396387328L) → 13
```

4. Игроки пытаются набрать очки, формируя слова, используя буквы из 6-буквенного скремблированного слова. Они выигрывают раунд, если им удается успешно расшифровать слово из 6 букв.

Создайте функцию, которая принимает в массив уже угаданных слов расшифрованное 6-буквенное слово и возвращает общее количество очков, набранных игроком в определенном раунде, используя следующую рубрику:

3-буквенные слова – это 1 очко
4-буквенные слова – это 2 очка
5-буквенные слова – это 3 очка
6-буквенные слова – это 4 очка + 50 пт бонуса (за расшифровку слова)
Помните, что недопустимые слова (слова, которые не могут быть сформированы из 6-буквенных расшифрованных слов) считаются 0 очками.

Пример:

```
totalPoints(["cat", "create", "sat"], "caster") → 2
// Since "create" is an invalid word.

totalPoints(["trance", "recant"], "recant") → 108
// Since "trance" and "recant" score 54 pts each.

totalPoints(["dote", "dotes", "toes", "set", "dot", "dots", "sted"],
"tossed") → 13
// Since 2 + 3 + 2 + 1 + 1 + 2 + 2 = 13
```

Примечание:
- Если 6-буквенное слово имеет несколько анаграмм, считайте каждую 6-буквенную расшифровку дополнительными 54 очками. Например, если слово arches, а игрок угадал arches и chaser, добавьте 108 очков для обоих слов.
- Вы можете играть в Текстовый Твист здесь: http://text-twist2.com

5. Создайте функцию, которая получает каждую пару чисел из массива, который суммирует до восьми, и возвращает его как массив пар (отсортированный по возрастанию).

Пример:

```
sumsUp([1, 2, 3, 4, 5]) → [[3, 5]]

sumsUp([1, 2, 3, 7, 9]) → [[1, 7]]

sumsUp([10, 9, 7, 2, 8]) → []

sumsUp([1, 6, 5, 4, 8, 2, 3, 7]) → [[2, 6], [3, 5], [1, 7]]
```

```
// [6, 2] first to complete the cycle (to sum up to 8)
// [5, 3] follows
// [1, 7] lastly
// the pair that completes the cycle is always found on the left
// [2, 6], [3, 5], [1, 7] sorted according to cycle completeness, then
pair-wise.
```

6. Какой процент вы можете набрать на тесте, который в одиночку снижает средний балл по классу на 5%? Учитывая массив оценок ваших одноклассников, создайте функцию, которая возвращает ответ. Округлите до ближайшего процента.

Пример:

```
takeDownAverage(["95%", "83%", "90%", "87%", "88%", "93%"]) → "54%"

takeDownAverage(["10%"]) → "0%"

takeDownAverage(["53%", "79%"]) → "51%"
```

7. Создайте функцию, которая будет шифровать и дешифровать сообщения с использованием шифра Цезаря. Шифр Цезаря – это метод шифрования, в котором каждая буква в сообщении сдвигается на фиксированное количество позиций в алфавите. Например, если сдвиг составляет 3 позиции, то буква 'A' будет зашифрована как 'D', 'B' как 'E' и так далее.

   Функция должна выполнять следующие действия:

   1. Определять режим работы: шифрование или дешифрование сообщения.
   2. Если пользователь хочет зашифровать сообщение, программа должна запросить само сообщение и сдвиг, на который нужно зашифровать текст.
   3. Если пользователь хочет дешифровать сообщение, программа должна запросить зашифрованное сообщение и сдвиг, чтобы расшифровать его.
   4. Обрабатывать сообщения только в верхнем регистре и оставлять другие символы (пробелы, цифры и специальные символы) без изменений.

Пример:

```
caesarCipher("encode", "hello world", 3) → " KHOOR ZRUOG"

caesarCipher(["decode", "almost last task!", 4]) → "EPQSWX PEWX XEWO!"
```

8. Создайте метод для рекурсивного вычисления количества различных способов как можно разместить k элементов из множества из n элементов без повторений. Это задача комбинаторики, и она часто используется в комбинаторных оптимизациях, таких как размещение задач на стеллажах или распределение ресурсов.

   Метод принимает два параметра, где n - количество элементов в множестве, а k - количество элементов, которые нужно разместить (n >= k) и рассчитывает количество размещений по формуле размещений без повторений: n! / (n - k)!

Пример:

```
setSetup(5, 3) → 60

setSetup(7, 3) → 210
```

9. В этой задаче цель состоит в том, чтобы вычислить, сколько времени сейчас в двух разных городах. Вам дается строка cityA и связанная с ней строка timestamp (time in cityA) с датой, отформатированной в полной нотации США, как в этом примере:

```
"July 21, 1983 23:01"
```

Вы должны вернуть новую метку времени с датой и соответствующим временем в cityB, отформатированную как в этом примере:

```
"1983-7-22 23:01"
```

Список данных городов и их смещения по Гринвичу (среднее время по Гринвичу) приведены в таблице ниже.

GMT	City
- 08:00	Los Angeles
- 05:00	New York
- 04:30	Caracas
- 03:00	Buenos Aires
00:00	London
+ 01:00	Rome
+ 03:00	Moscow
+ 03:30	Tehran
+ 05:30	New Delhi
+ 08:00	Beijing
+ 10:00	Canberra

Пример:

```
timeDifference("Los Angeles", "April 1, 2011 23:23", "Canberra") ➞ "2011-4-2 17:23"
// Can be a new day.

timeDifference("London", "July 31, 1983 23:01", "Rome") ➞ "1983-8-1 00:01"
// Can be a new month.

timeDifference("New York", "December 31, 1970 13:40", "Beijing") ➞ "1971-1-1 02:40"
// Can be a new year.
```

Примечание:
- Обратите внимание на часы и минуты, ведущий 0 необходим в возвращаемой метке времени, когда они представляют собой одну цифру.
- Обратите внимание на города с получасовыми смещениями.

10. Новое число – это число, которое не является перестановкой любого меньшего числа. 869 – это не новое число, потому что это просто перестановка меньших чисел, 689 и 698. 509 – это новое число, потому что оно не может быть образовано перестановкой любого меньшего числа (ведущие нули не допускаются).

Напишите функцию, которая принимает неотрицательное целое число и возвращает true, если целое число является новым числом, и false, если это не так.

Пример:

```
isNew(3) ➞ true

isNew(30) ➞ true

isNew(321) ➞ false

isNew(123) ➞ true
```

6/6

1. Создайте функцию, которая принимает две строки. Первая строка содержит предложение, содержащее буквы второй строки в последовательной последовательности, но в другом порядке. Скрытая анаграмма должна содержать все буквы, включая дубликаты, из второй строки в любом порядке и не должна содержать никаких других букв алфавита.
   Напишите функцию, чтобы найти анаграмму второй строки, вложенную где-то в первую строку. Вы должны игнорировать регистр символов, любые пробелы и знаки препинания и возвращать анаграмму в виде строчной строки без пробелов или знаков препинания.

   Пример:

```
hiddenAnagram(["My world evolves in a beautiful space called Tesh.",
"sworn love lived"]) ➞ "worldevolvesin"
// The sequence "world evolves in" is a perfect anagram of "sworn love
lived".

hiddenAnagram("An old west action hero actor", "Clint Eastwood") ➞
"noldwestactio"
// The sequence "n old west actio" is a perfect anagram of "Clint
Eastwood".

hiddenAnagram("Mr. Mojo Rising could be a song title", "Jim Morrison") ➞
"mrmojorisin"
// The sequence "Mr. Mojo Risin" ignoring the full stop, is a perfect
// anagram of "Jim Morrison".

hiddenAnagram("Banana? margaritas", "ANAGRAM") ➞ "anamarg"
// The sequence "ana? marg" ignoring "?" is a perfect anagram of
"Anagram".

hiddenAnagram("D  e b90it->?$ (c)a r...d,,#~", "bad credit") ➞
"debitcard"
// When all spaces, numbers and puntuation marks are removed
// from the whole phrase, the remaining characters form the sequence
// "Debitcard" which is a perfect anagram of "bad credit".

hiddenAnagram("Bright is the moon", "Bongo mirth") ➞ "notfound"
// The words "Bright moon" are an anagram of "bongo mirth" but they are
// not a continuous alphabetical sequence because the words "is the" are
in
// between. Hence the negative result, "notfound" is returned.
```

   Примечание:

   - Совершенная анаграмма содержит все буквы оригинала в любом порядке, ни больше, ни меньше.
   - Если в предложении нет скрытой анаграммы, верните "notfound".
   - Как и в приведенных выше примерах, скрытая анаграмма может начинаться или заканчиваться частично через слово и/или охватывать несколько слов и может содержать знаки препинания и другие не-альфа-символы.
   - Игнорируйте регистр символов и любые встроенные не-альфа-символы.
   - Если в предложении больше 1 результата, верните ближайший к началу.

2. Напишите функцию, которая возвращает массив строк, заполненных из срезов символов n-длины данного слова (срез за другим, в то время как n-длина применяется к слову).

Пример:

```
collect("intercontinentalisationalism", 6)
➞ ["ationa", "interc", "ntalis", "ontine"]

collect("strengths", 3)
➞ ["eng", "str", "ths"]

collect("pneumonoultramicroscopicsilicovolcanoconiosis", 15)
➞ ["croscopicsilico", "pneumonoultrami", "volcanoconiosis"]
```

Примечания:

- Убедитесь, что результирующий массив лексикографически упорядочен.
- Возвращает пустой массив, если заданная строка меньше n.
- Ожидается, что вы решите эту задачу с помощью рекурсивного подхода.

3. В шифре Nico кодирование осуществляется путем создания цифрового ключа и присвоения каждой буквенной позиции сообщения с помощью предоставленного ключа.

Создайте функцию, которая принимает два аргумента, message и key, и возвращает закодированное сообщение.

Существуют некоторые вариации правил шифрования. Одна из версий правил шифрования изложена ниже:

```
message = "mubashirhassan"
key = "crazy"

nicoCipher(message, key) ➞ "bmusarhiahass n"
```

Шаг 1: Назначьте числа отсортированным буквам из ключа:

```
"crazy" = 23154
```

Шаг 2: Назначьте номера буквам данного сообщения:

```
2 3 1 5 4
---------
m u b a s
h i r h a
s s a n
```

Шаг 3: Сортировка столбцов по назначенным номерам:

```
1 2 3 4 5
---------
b m u s a
r h i a h
a s s   n
```

Шаг 4: Верните закодированное сообщение по строкам:

```
eMessage = "bmusarhiahass n"
```

Пример:

```
nicoCipher("myworldevolvesinhers", "tesh") ➞ "yowmledrovlvsnieesrh"

nicoCipher("andiloveherso", "tesha") ➞ "lnidaevheo s or"

nicoCipher("mubashirhassan", "crazy") ➞ "bmusarhiahass n"

nicoCipher("edabitisamazing", "matt") ➞ "deabtiismaaznig "

nicoCipher("iloveher", "612345") ➞ "lovehir    e"
```

4. Создайте метод, который принимает массив arr и число n и возвращает массив из двух целых чисел из arr, произведение которых равно числу n следующего вида:

```
[value_at_lower_index, value_at_higher_index]
```

Убедитесь, что вы возвращаете массив из двух целых чисел, который точно делит n (произведение n) и что индексы совпадают с указанным выше форматом. Таким образом, сортировка значений основана на восходящих индексах.

Пример:

```
twoProduct([1, 2, 3, 9, 4, 5, 15], 45) ➞ [9, 5]
// at index 5 which has the value 5 is a full match
// to the value at index 3 which is 9
// the closest gap between indices that equates
// to the product which is 45

twoProduct([1, 2, 3, 9, 4, 15, 3, 5], 45) ➞ [3, 15]
// at index 5 which has the value 15 is a full match
// to the value at index 2 which is 3
// the closest gap between indices that equates
// to the product which is 45

twoProduct([1,  2, -1,  4,  5,  6,  10, 7], 20) ➞ [4, 5]
// at index 4 which has the value 5 is a full match
// to the value at index 3 which is 4
// a full match can only be achieved if you've found the multiplier at an
// index lower than the current index, thus, 5 (@ index 4) has a pair
lower
// than its index which is the value 4 (@ index 3), so, 5*4 = 20, in which
5
// has a higher index (4) than 4 which has an index value of 3

twoProduct([1, 2, 3, 4, 5,  6, 7, 8, 9, 10], 10) ➞ [2, 5]

twoProduct([100, 12, 4, 1, 2], 15) ➞ []
```

Примечание:
- Дубликатов не будет.
- Возвращает пустой массив, если совпадение не найдено.
- Всегда считайте, что пара рассматриваемого элемента (текущий элемент, на который указывали во время поиска) находится слева от него.
- Массив может иметь несколько решений (произведений n), поэтому возвращайте первое найденное полное совпадение (как описано выше).

5. Создайте рекурсивную функцию, которая проверяет, является ли число точной верхней границей факториала n. Если это так, верните массив точной факториальной границы и n, или иначе, пустой массив.

Пример:

```
isExact(6) → [6, 3]

isExact(24) → [24, 4]

isExact(125) → []

isExact(720) → [720, 6]

isExact(1024) → []

isExact(40320) → [40320, 8]
```

6. Деление на дробь часто приводит к бесконечно повторяющейся десятичной дроби.

```
1/3=.3333333...  1/7=.142857142857...
```

Создайте функцию, которая принимает десятичную дробь в строковой форме с повторяющейся частью в круглых скобках и возвращает эквивалентную дробь в строковой форме и в наименьших членах.

Пример:

```
fractions("0.(6)") → "2/3"

fractions("1.(1)") → "10/9"

fractions("3.(142857)") → "22/7"

fractions("0.19(2367)") → "5343/27775"

fractions("0.1097(3)") → "823/7500"
```

7. В этой задаче преобразуйте строку в серию слов (или последовательности символов), разделенных одним пробелом, причем каждое слово имеет одинаковую длину, заданную первыми 15 цифрами десятичного представления числа Пи:

```
3.14159265358979
```

Если строка содержит больше символов, чем общее количество, заданное суммой цифр Пи, неиспользуемые символы отбрасываются, и вы будете использовать только те, которые необходимы для формирования 15 слов.

```
String =
"HOWINEEDADRINKALCOHOLICINNATUREAFTERTHEHEAVYLECTURESINVOLVINGQUANTUMMECHA
NICSANDALLTHESECRETSOFTHEUNIVERSE"

Pi String = "HOW I NEED A DRINK ALCOHOLIC IN NATURE AFTER THE HEAVY
LECTURES INVOLVING QUANTUM MECHANICS"

// Every word has the same length of the digit of Pi found at the same
index ?
```

```
// "HOW" = 3, "I" = 1, "NEED" = 4, "A" = 1, "DRINK" = 5
// "ALCOHOLIC" = 9, "IN" = 2, "NATURE" = 6, "AFTER" = 5
// "THE" = 3, "HEAVY" = 5, "LECTURES" = 8, "INVOLVING" = 9
// "QUANTUM" = 7, "MECHANICS" = 9
// 3.14159265358979
```

Также, если строка содержит меньше символов, чем общее количество, заданное суммой цифр Пи, в любом случае вы должны соблюдать последовательность цифр для получения слов.

```
String = "FORALOOP"

Pi String = "FOR A LOOP"

// Every word has the same length of the digit of Pi found at the same
index ?
// "FOR" = 3, "A" = 1, "LOOP" = 4
// 3.14
```

Если буквы, содержащиеся в строке, не совпадают в точности с цифрами, в этом случае вы будете повторять последнюю букву, пока слово не будет иметь правильную длину.

```
String = "CANIMAKEAGUESSNOW"

Pi String = "CAN I MAKE A GUESS NOWWWWWWW"

// Every word has the same length of the digit of Pi found at the same
index ?
// "CAN" = 3, "I" = 1, "MAKE" = 4, "A" = 1, "GUESS" = 5, "NOW" = 3
// 3.14153 (Wrong!)
// The length of the sixth word "NOW" (3)...
// ...doesn't match the sixth digit of Pi (9)
// The last letter "W" will be repeated...
// ...until the length of the word will match the digit

// "CAN" = 3, "I" = 1, "MAKE" = 4, "A" = 1, "GUESS" = 5, "NOWWWWWWW" = 9
// 3.14159 (Correct!)
```

Если данная строка пуста, должна быть возвращена пустая строка.

Учитывая строку txt, реализуйте функцию, которая возвращает ту же строку, отформатированную в соответствии с приведенными выше инструкциями.

Пример:

```
pilish_string("33314444") ➞ "333 1 4444"
// 3.14

pilish_string("TOP") ➞ "TOP"
// 3

pilish_string("X")➞ "XXX"
// "X" has to match the same length of the first digit (3)
// The last letter of the word is repeated

pilish_string("")➞ ""
```

Примечание:

- Эта задача представляет собой упрощенную концепцию, вдохновленную Пилишем, своеобразным типом ограниченного письма, в котором используются все известные возможные цифры Пи. Потенциально бесконечный текст может быть написан с использованием знаков препинания и набора дополнительных правил, которые позволяют избежать длинных последовательностей маленьких цифр, заменяя их словами больше 9 букв и делая так, чтобы композиция выглядела более похожей на стихотворение вольным стихом.
- Точка, отделяющая целую часть числа Пи от десятичной, не должна учитываться в функции: она присутствует в инструкциях и примерах только для удобства чтения.

8. Создайте функцию, которая будет вычислять результат математических выражений, предоставленных в виде строки.

   Реализуйте алгоритм, который разбирает строку и вычисляет результат выражения, учитывая приоритет операций, скобки и т. д. Математические операции, которые нужно поддерживать, включают в себя сложение, вычитание, умножение, деление и скобки. Обработайте ошибки, такие как деление на ноль или неправильно введенное выражение, и верните соответствующее сообщение об ошибке.

Пример:

```
generateNonconsecutive("3 + 5 * (2 - 6)") ➞ -17

generateNonconsecutive("6 - 18 / (-1 + 4)") ➞ 0
```

9. Шерлок считает строку действительной, если все символы строки встречаются одинаковое количество раз. Также допустимо, если он может удалить только 1 символ из 1 индекса в строке, а остальные символы будут встречаться одинаковое количество раз. Для данной строки str определите, действительна ли она. Если да, верните «ДА», в противном случае верните «НЕТ».

   Например, если str = "abc", строка действительна, потому что частота символов у всех одинакова. Если str = "abcc", строка также действительна, потому что мы можем удалить 1 "c" и оставить по одному символу каждого символа в строке. Однако, если str = "abccc", строка недействительна, поскольку удаление одного символа не приводит к одинаковой частоте символов.

   Пример:

```
isValid("aabbcd") ➞ "NO"
// We would need to remove two characters, both c and d  -> aabb or a and
b -> abcd, to make it valid.
// We are limited to removing only one character, so it is invalid.

isValid("aabbccddeefghi") ➞ "NO"
// Frequency counts for the letters are as follows:
// {"a": 2, "b": 2, "c": 2, "d": 2, "e": 2, "f": 1, "g": 1, "h": 1, "i":
1}
// There are two ways to make the valid string:
// Remove 4 characters with a frequency of 1: {f, g, h, i}.
// Remove 5 characters of frequency 2: {a, b, c, d, e}.
// Neither of these is an option.

isValid("abcdefghhgfedecba") ➞ "YES"
// All characters occur twice except for e which occurs 3 times.
```

```
// We can delete one instance of e to have a valid string.
```

10. Создайте функцию, которая будет находить наибольшую общую подпоследовательность (LCS) для двух строк. LCS – это самая длинная последовательность символов, которая встречается как подпоследовательность в обеих строках. Эта задача требует понимания алгоритма динамического программирования для нахождения наибольшей общей подпоследовательности и его эффективной реализации.

Пример:

```
findLCS("abcd", "bd") → "bd"

findLCS("aggtab", "gxtxamb") → "gtab"
```