1/6

1. Напишите функцию, которая принимает целое число минут и преобразует его в секунды.

Пример:

```
convert(5) → 300

convert(3) → 180

convert(2) → 120
```

2. Вы подсчитываете очки за баскетбольный матч, учитывая количество забитых 2-х и 3-х очков, находите окончательные очки для команды и возвращаете это значение.

Пример:

```
points(13, 12) → 62

points(17, 12) → 70

points(38, 8) → 100
```

3. Создайте функцию, которая принимает количество побед, ничьих и поражений и вычисляет количество очков, набранных футбольной командой на данный момент.

выигрыш – получают 3 очка
ничья – получают 1 очко
проигрыш – получают 0 очков

Пример:

```
footballPoints(3, 4, 2) → 13

footballPoints(5, 0, 2) → 15

footballPoints(0, 0, 1) → 0
```

4. Создайте функцию, которая возвращает true, если целое число равномерно делится на 5, и false в противном случае.

Пример:

```
divisibleByFive(5) → true

divisibleByFive(-55) → true

divisibleByFive(37) → false
```

5. В Java есть логический оператор &&. Оператор && принимает два логических значения и возвращает true, если оба значения истинны.

Рассмотрим a && b:

a проверяется, является ли оно истинным или ложным.
Если a равно false, возвращается false.

b проверяется, является ли оно истинным или ложным.
Если b имеет значение false, возвращается значение false.
В противном случае возвращается true (поскольку и a, и b, следовательно, истинны).
Оператор && вернет true только для true && true.

Создайте функцию с помощью оператора&&.

Пример:

```
and(true, false) → false

and(true, true) → true

and(false, true) → false

and(false, false) → false
```

6. У меня есть ведро с большим количеством темно-синей краски, и я хотел бы покрасить как можно больше стен. Создайте функцию, которая возвращает количество полных стен, которые я могу покрасить, прежде чем мне нужно будет отправиться в магазины, чтобы купить еще.

n - это количество квадратных метров, которые я могу нарисовать.
w и h-это ширина и высота одной стены в метрах.

Пример:

```
howManyWalls(54, 1, 43) → 1

howManyWalls(46, 5, 4) → 2

howManyWalls(100, 4, 5) → 5

howManyWalls(10, 15, 12) → 0

howManyWalls(41, 3, 6) → 2
```

Примечание:
- Не считайте стену, если я не успею закончить покраску до того, как у меня закончится краска.
- Все стены будут иметь одинаковые размеры.
- Все числа будут целыми положительными.

7. Исправьте код, чтобы решить эту задачу (только синтаксические ошибки). Посмотрите на приведенные ниже примеры, чтобы получить представление о том, что должна делать эта функция.

```
squared(5) → 25

squared(9) → 81

squared(100) → 10000
```

Код:

```
public class Challenge {
     public static int squaed(int b) {
```

```
            return a * a
    }
}
```

8. Создайте функцию, которая принимает три аргумента prob, prize, pay и возвращает true, если prob * prize > pay; в противном случае возвращает false.

Чтобы проиллюстрировать:

```
profitableGamble(0.2, 50, 9)
```

... должно давать true, так как чистая прибыль равна 1 (0.2 * 50 - 9), и 1 > 0.

Пример:

```
profitableGamble(0.2, 50, 9) → true

profitableGamble(0.9, 1, 2) → false

profitableGamble(0.9, 3, 2) → true
```

9. Создайте метод, который возвращает количество кадров, показанных за заданное количество минут для определенного FPS.

Пример:

```
frames(1, 1) → 60

frames(10, 1) → 600

frames(10, 25) → 15000
```

10. Создайте функцию, которая будет работать как оператор модуля % без использования оператора модуля. Оператор модуля-это способ определения остатка операции деления. Вместо того чтобы возвращать результат деления, операция по модулю возвращает остаток целого числа.

Пример:

```
mod(5, 2) → 1

mod(218, 5) → 3

mod(6, 3) → 0
```

2/6

1. Теша шел по прямой улице, по обеим сторонам которой стояло ровно n одинаковых домов. Номера домов на улице выглядят так:

```
1 |   | 6

3 |   | 4

5 |   | 2
```

Она заметила, что четные дома увеличиваются справа, а нечетные уменьшаются слева.

Создайте функцию, которая принимает номер дома и длину улицы n и возвращает номер дома на противоположной стороне.

Пример:

```
oppositeHouse(1, 3) → 6

oppositeHouse(2, 3) → 5

oppositeHouse(3, 5) → 8

oppositeHouse(5, 46) → 88
```

2. Создайте метод, который принимает строку (имя и фамилию человека) и возвращает строку с заменой имени и фамилии.

Пример:

```
nameShuffle("Donald Trump") → "Trump Donald"

nameShuffle("Rosie O'Donnell") → "O'Donnell Rosie"

nameShuffle("Seymour Butts") → "Butts Seymour"
```

3. Создайте функцию, которая принимает два аргумента: исходную цену и процент скидки в виде целых чисел и возвращает конечную цену после скидки.

Пример:

```
discount(1500, 50) → 750

discount(89, 20) → 71.2

discount(100, 75) → 25
```

4. Создайте функцию, которая принимает массив и возвращает разницу между наибольшим и наименьшим числами.

Пример:

```
differenceMaxMin([10, 4, 1, 4, -10, -50, 32, 21]) → 82
// Smallest number is -50, biggest is 32.

differenceMaxMin([44, 32, 86, 19]) → 67
```

```
// Smallest number is 19, biggest is 86.
```

5. Создайте функцию, которая принимает три целочисленных аргумента (a, b, c) и возвращает количество целых чисел, имеющих одинаковое значение.

Пример:

```
equal(3, 4, 3) → 2

equal(1, 1, 1) → 3

equal(3, 4, 1) → 0
```

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"

reverse("Edabit is really helpful!") → "!lufpleh yllaer si tibadE"
```

7. Вы наняли трех программистов и (надеюсь) платите им. Создайте функцию, которая принимает три числа (почасовая заработная плата каждого программиста) и возвращает разницу между самым высокооплачиваемым программистом и самым низкооплачиваемым.

Пример:

```
programmers(147, 33, 526) → 493

programmers(33, 72, 74) → 41

programmers(1, 5, 9) → 8
```

8. Создайте функцию, которая принимает строку, проверяет, имеет ли она одинаковое количество x и o и возвращает либо true, либо false.

Правила:
- Возвращает логическое значение (true или false).
- Верните true, если количество x и o одинаковы.
- Верните false, если они не одинаковы.
- Строка может содержать любой символ.
- Если "x" и "o" отсутствуют в строке, верните true.

Пример:

```
getXO("ooxx") → true

getXO("xooxx") → false

getXO("ooxXm") → true
// Case insensitive.

getXO("zpzpzpp") → true
```

```
// Returns true if no x and o.

getXO("zzoo") → false
```

9. Напишите функцию, которая находит слово "бомба" в данной строке. Ответьте "ПРИГНИСЬ!", если найдешь, в противном случае:"Расслабься, бомбы нет".

Пример:

```
bomb("There is a bomb.") → "DUCK!"

bomb("Hey, did you think there is a BOMB?") → "DUCK!"

bomb("This goes boom!!!") → "Relax, there's no bomb."
```

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

10. Возвращает true, если сумма значений ASCII первой строки совпадает с суммой значений ASCII второй строки, в противном случае возвращает false.

Пример:

```
sameAscii("a", "a") → true

sameAscii("AA", "B@") → true

sameAscii("EdAbIt", "EDABIT") → false
```

3/6

1. Учитывая массив городов и населения, верните массив, в котором все население округлено до ближайшего миллиона.

Пример:

```
millionsRounding([
  ["Nice", 942208],
  ["Abu Dhabi", 1482816],
  ["Naples", 2186853],
  ["Vatican City", 572]
]) ➞ [
  ["Nice", 1000000],
  ["Abu Dhabi", 1000000],
  ["Naples", 2000000],
  ["Vatican City", 0]
]

millionsRounding([
  ["Manila", 13923452],
  ["Kuala Lumpur", 7996830],
  ["Jakarta", 10770487]
]) ➞ [
  ["Manila", 14000000],
  ["Kuala Lumpur", 8000000],
  ["Jakarta", 11000000]
]
```

2. Учитывая самую короткую сторону треугольника 30° на 60° на 90°, вы должны найти другие 2 стороны (верните самую длинную сторону, сторону средней длины).

Пример:

```
otherSides(1) ➞ [2.0, 1.73]

otherSides(12) ➞ [24.0, 20.0]

otherSides(2) ➞ [4.0, 3.46]

otherSides(3) ➞ [6.0, 5.2]
```

Примечание:
- треугольники 30° на 60° на 90° всегда следуют этому правилу, скажем, самая короткая длина стороны равна x единицам, гипотенуза будет равна 2 единицам, а другая сторона будет равна x * root3 единицам.
- Результаты тестов округляются до 2 знаков после запятой.
- Верните результат в виде массива.

3. Создайте функцию, имитирующую игру "камень, ножницы, бумага". Функция принимает входные данные обоих игроков (камень, ножницы или бумага), первый параметр от первого игрока, второй от второго игрока. Функция возвращает результат как таковой:

"Игрок 1 выигрывает"
"Игрок 2 выигрывает"

"НИЧЬЯ" (если оба входа одинаковы)
Правила игры камень, ножницы, бумага, если не известны:

Оба игрока должны сказать одновременно "камень", "бумага" или "ножницы".
Камень бьет ножницы, бумага бьет камень, ножницы бьют бумагу.

Пример:

```
rps("rock", "paper") → "Player 2 wins"

rps("paper", "rock") → "Player 1 wins"

rps("paper", "scissors") → "Player 2 wins"

rps("scissors", "scissors") → "TIE"

rps("scissors", "paper") → "Player 1 wins"
```

4. Идет великая война между четными и нечетными числами. Многие уже погибли в этой войне, и ваша задача-положить этому конец. Вы должны определить, какая группа суммируется больше: четная или нечетная. Выигрывает большая группа.

Создайте функцию, которая берет массив целых чисел, суммирует четные и нечетные числа отдельно, а затем возвращает разницу между суммой четных и нечетных чисел.

Пример:

```
warOfNumbers([2, 8, 7, 5]) → 2
// 2 + 8 = 10
// 7 + 5 = 12
// 12 is larger than 10
// So we return 12 - 10 = 2

warOfNumbers([12, 90, 75]) → 27

warOfNumbers([5, 9, 45, 6, 2, 7, 34, 8, 6, 90, 5, 243]) → 168
```

5. Учитывая строку, создайте функцию для обратного обращения. Все буквы в нижнем регистре должны быть прописными, и наоборот.

Пример:

```
reverseCase("Happy Birthday") → "hAPPY bIRTHDAY"

reverseCase("MANY THANKS") → "many thanks"

reverseCase("sPoNtAnEoUs") → "SpOnTaNeOuS"
```

6. Создайте функцию, которая принимает строку из одного слова и выполняет следующие действия:

Конкатенирует `inator` до конца, если слово заканчивается согласным, в противном случае вместо него конкатенирует `-inator`
Добавляет длину слова исходного слова в конец, снабженный '000'.
Пример:

```
inatorInator("Shrink") → "Shrinkinator 6000"
```

```
inatorInator("Doom") → "Doominator 4000"

inatorInator("EvilClone") → "EvilClone-inator 9000"
```

7. Напишите функцию, которая принимает три измерения кирпича: высоту(a), ширину(b) и глубину(c) и возвращает true, если этот кирпич может поместиться в отверстие с шириной(w) и высотой(h).

Пример:

```
doesBrickFit(1, 1, 1, 1, 1) → true

doesBrickFit(1, 2, 1, 1, 1) → true

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

Примечание:
- Вы можете повернуть кирпич любой стороной к отверстию.
- Мы предполагаем, что кирпич подходит, если его размеры равны размерам отверстия (то есть размер кирпича должен быть меньше или равен размеру отверстия, а не строго меньше).
- Нельзя класть кирпич под неортогональным углом.

8. Напишите функцию, которая принимает топливо (литры), расход топлива (литры/100 км), пассажиров, кондиционер (логическое значение) и возвращает максимальное расстояние, которое может проехать автомобиль.

топливо-это количество литров топлива в топливном баке.
Расход топлива-это базовый расход топлива на 100 км (только с водителем внутри).
Каждый дополнительный пассажир увеличивает базовый расход топлива на 5%.
Если кондиционер включен, то его общий (не базовый) расход топлива увеличивается на 10%.

Пример:

```
totalDistance(70.0, 7.0, 0, false) → 1000.0

totalDistance(36.1, 8.6, 3, true) → 331.83

totalDistance(55.5, 5.5, 5, false) → 807.3
```

9. Создайте функцию, которая принимает массив чисел и возвращает среднее значение (average) всех этих чисел.

Пример:

```
mean([1, 0, 4, 5, 2, 4, 1, 2, 3, 3, 3]) → 2.55

mean([2, 3, 2, 3]) → 2.50

mean([3, 3, 3, 3, 3]) → 3.00
```

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

Пример:

```
parityAnalysis(243) ➞ true
// 243 is odd and so is 9 (2 + 4 + 3)

parityAnalysis(12) ➞ false
// 12 is even but 3 is odd (1 + 2)

parityAnalysis(3) ➞ true
// 3 is odd and 3 is odd and 3 is odd (3)
```

4/6

1. Создайте функцию, которая принимает массив чисел и возвращает "Бум!", если в массиве появляется цифра 7. В противном случае верните "в массиве нет 7".

Пример:

```
sevenBoom([1, 2, 3, 4, 5, 6, 7]) → "Boom!"
// 7 contains the number seven.

sevenBoom([8, 6, 33, 100]) → "there is no 7 in the array"
// None of the items contain 7 within them.

sevenBoom([2, 55, 60, 97, 86]) → "Boom!"
// 97 contains the number seven.
```

2. Создайте функцию, которая определяет, могут ли элементы в массиве быть переупорядочены, чтобы сформировать последовательный список чисел, где каждое число появляется ровно один раз.

```
cons([5, 1, 4, 3, 2]) → true
// Can be re-arranged to form [1, 2, 3, 4, 5]

cons([5, 1, 4, 3, 2, 8]) → false

cons([5, 6, 7, 8, 9, 9]) → false
// 9 appears twice
```

3. lPaeesh le pemu mnxit ehess rtnisg! О, извините, это должно было быть: Пожалуйста, помогите мне распутать эти строки!

Каким-то образом все строки перепутались, каждая пара символов поменялась местами. Помоги отменить это, чтобы снова понять строки.

Пример:

```
unmix("123456") → "214365"

unmix("hTsii  s aimex dpus rtni.g") → "This is a mixed up string."

unmix("badce") → "abcde"
```

4. Создать функцию, которая преобразует предложения, заканчивающиеся несколькими вопросительными знаками ? или восклицательными знаками ! в предложение, заканчивающееся только одним, без изменения пунктуации в середине предложений.

Пример:

```
noYelling("What went wrong?????????") → "What went wrong?"

noYelling("Oh my goodness!!!") → "Oh my goodness!"

noYelling("I just!!! can!!! not!!! believe!!! it!!!") → "I just!!!
can!!! not!!! believe!!! it!"
// Only change repeating punctuation at the end of the sentence.
```

```
noYelling("Oh my goodness!") ➞ "Oh my goodness!"
// Do not change sentences where there exists only one or zero
exclamation marks/question marks.

noYelling("I just cannot believe it.") ➞ "I just cannot believe it."
```

Примечание:
- Меняйте только окончательную пунктуацию - оставляйте восклицательные или вопросительные знаки в середине предложения неизменными (см. Третий пример).
- Не беспокойтесь о смешанной пунктуации (нет случаев, которые заканчиваются чем-то вроде ?!??!).
- Сохраняйте предложения, в которых нет вопросительных/восклицательных знаков, одинаковыми.

5. Создайте функцию, которая заменяет все x в строке следующими способами:

Замените все x на "cks", ЕСЛИ ТОЛЬКО:
Слово начинается с "x", поэтому замените его на "z".
Слово-это просто буква "x", поэтому замените ее на " cks ".

Пример:

```
xPronounce("Inside the box was a xylophone") ➞ "Inside the bocks was a
zylophone"

xPronounce("The x ray is excellent") ➞ "The ecks ray is eckscellent"

xPronounce("OMG x box unboxing video x D") ➞ "OMG ecks bocks unbocksing
video ecks D"
```

Примечание:
- Все X являются строчными.

6. Учитывая массив целых чисел, верните наибольший разрыв между отсортированными элементами массива.

Например, рассмотрим массив:

```
[9, 4, 26, 26, 0, 0, 5, 20, 6, 25, 5]
```

... в котором после сортировки массив становится:

```
[0, 0, 4, 5, 5, 6, 9, 20, 25, 26, 26]
```

.. так что теперь мы видим, что самый большой разрыв в массиве находится между 9 и 20, что равно 11.

Пример:

```
largestGap([9, 4, 26, 26, 0, 0, 5, 20, 6, 25, 5]) ➞ 11
// After sorting: [0, 0, 4, 5, 5, 6, 9, 20, 25, 26, 26]
// Largest gap between 9 and 20 is 11

largestGap([14, 13, 7, 1, 4, 12, 3, 7, 7, 12, 11, 5, 7]) ➞ 4
// After sorting: [1, 3, 4, 5, 7, 7, 7, 7, 11, 12, 12, 13, 14]
// Largest gap between 7 and 11 is 4
```

```
largestGap([13, 3, 8, 5, 5, 2, 13, 6, 14, 2, 11, 4, 10, 8, 1, 9]) → 2
// After sorting: [1, 2, 2, 3, 4, 5, 5, 6, 8, 8, 9, 10, 11, 13, 13, 14]
// Largest gap between 6 and 8 is 2
```

7. Это вызов обратного кодирования. Обычно вам дают явные указания о том, как создать функцию. Здесь вы должны сгенерировать свою собственную функцию, чтобы удовлетворить соотношение между входами и выходами.

Ваша задача состоит в том, чтобы создать функцию, которая при подаче входных данных ниже производит показанные примеры выходных данных.

Пример:

```
832 → 594

51 → 36

7977 → 198

1 → 0

665 → 99

149 → 0
```

8. Создайте функцию, которая принимает предложение в качестве входных данных и возвращает наиболее распространенную последнюю гласную в предложении в виде одной символьной строки.

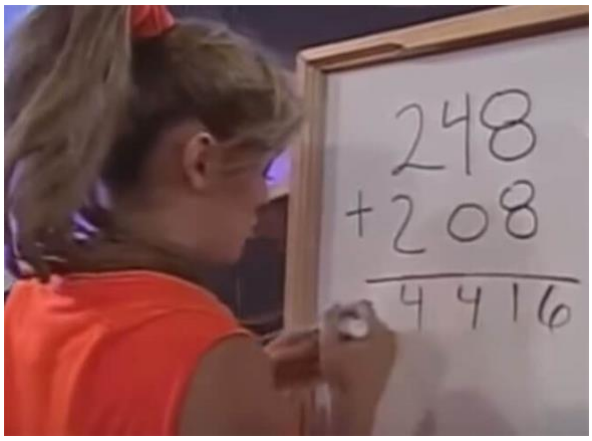
Пример:

```
commonLastVowel("Hello World!") → "o"

commonLastVowel("Watch the characters dance!") → "e"

commonLastVowel("OOI UUI EEI AAI") → "i"
```

9. Для этой задачи забудьте, как сложить два числа вместе. Лучшее объяснение того, что нужно сделать для этой функции, - это этот мем:



Пример:

```
memeSum(26, 39) → 515
// 2+3 = 5, 6+9 = 15
```

```
// 26 + 39 = 515

memeSum(122, 81) ➞ 1103
// 1+0 = 1, 2+8 = 10, 2+1 = 3
// 122 + 81 = 1103

memeSum(1222, 30277) ➞ 31499
```

10. Создайте функцию, которая удалит все повторяющиеся символы в слове, переданном этой функции. Не просто последовательные символы, а символы, повторяющиеся в любом месте строки.

Пример:

```
unrepeated("teshahset") ➞ "tesha"

unrepeated("hello") ➞ "helo"

unrepeated("aaaaa") ➞ "a"

unrepeated("WWE!!!") ➞ "WE!"

unrepeated("call 911") ➞ "cal 91"
```

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
```

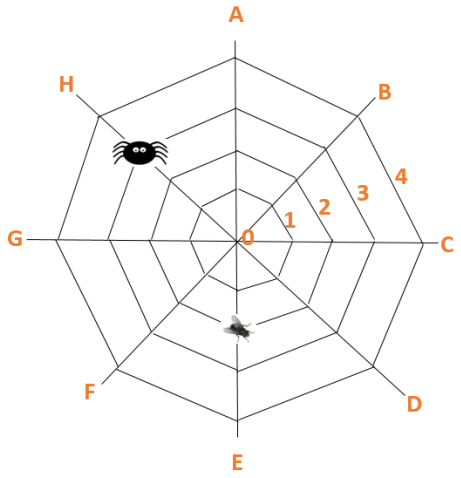
2. Паутина определяется кольцами, пронумерованными от 0 до 4 от центра, и радиалами, помеченными по часовой стрелке сверху как A-H.

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. Мы могли бы это устроить:

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. В 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. Последовательный прогон-это список соседних последовательных целых чисел. Этот список может быть как увеличивающимся, так и уменьшающимся. Создайте функцию, которая принимает массив чисел и возвращает длину самого длинного последовательного запуска.

```
longestRun([1, 2, 3, 5, 6, 7, 8, 9]) → 5
// Two consecutive runs: [1, 2, 3] and [5, 6, 7, 8, 9] (longest).
```

Пример:

```
longestRun([1, 2, 3, 10, 11, 15]) → 3
// Longest consecutive-run: [1, 2, 3].
```

```
longestRun([5, 4, 2, 1]) → 2
// Longest consecutive-run: [5, 4] and [2, 1].

longestRun([3, 5, 7, 10, 15]) → 1
// No consecutive runs, so we return 1.
```

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. Учитывая предложение с числами, представляющими расположение слова, встроенного в каждое слово, верните отсортированное предложение.

```
rearrange("Tesh3 th5e 1I lov2e way6 she7 j4ust i8s.") → "I love Tesh
just the way she is."

rearrange("the4 t3o man5 Happ1iest of6 no7 birt2hday steel8!") →
"Happiest birthday to the man of no steel!"

rearrange("is2 Thi1s T4est 3a") → "This is a Test"

rearrange("4of Fo1r pe6ople g3ood th5e the2") → "For the good of the
people"

rearrange(" ") → ""
```

8. Напишите функцию, которая делает первое число как можно больше, меняя его цифры на цифры во втором числе.

```
maxPossible(9328, 456) → 9658
// 9658 is the largest possible number built from swaps from 456.
// 3 replaced with 6 and 2 replaced with 5.
```

Пример:

```
maxPossible(523, 76) → 763

maxPossible(9132, 5564) → 9655

maxPossible(8732, 91255) → 9755
```

Примечание:
- Каждая цифра во втором числе может быть использована только один раз.
- Можно использовать ноль для всех цифр второго числа.

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") ➞ "noutfond"
// 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, "noutfond" is returned.
```

   Примечание:

   - Совершенная анаграмма содержит все буквы оригинала в любом порядке, ни больше, ни меньше.
   - Если в предложении нет скрытой анаграммы, верните "noutfond".
   - 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, 3], 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]
```

Примечание:
- Ожидается, что вы решите эту задачу с помощью рекурсии.

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 =
"HOWINEEDADRINKALCOHOLICINNATUREAFTERTHEHEAVYLECTURESINVOLVINGQUANTUMME
CHANICSANDALLTHESECRETSOFTHEUNIVERSE"
```

```
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 для генерации всех непоследовательных двоичных строк, где непоследовательные определяется как строка, в которой нет последовательных строк, и где n определяет длину каждой двоичной строки.

Пример:
```
generateNonconsecutive(1) ➞ "0 1"

generateNonconsecutive(2) ➞ "00 01 10"

generateNonconsecutive(3) ➞ "000 001 010 100 101"

generateNonconsecutive(4) ➞ "0000 0001 0010 0100 0101 1000 1001 1010"
```

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 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.
```