Использование PYTHON для решения заданий КЕГЭ

Задание 24, 25

Задание 24.
Обработка символьных строк

- Чтение данных из файла

```
with open("24.txt", "r") as F:
    str = F.readline()
```

```
F = open("24.txt")
str = F.readline()
```

```
str = open("24.txt").read()
```

```
str = open("24.txt").readline()
```

Типы задач

- Нахождение последовательности (цепочки) символов максимальной длины
- Нахождение нескольких последовательностей символов максимальной длины
- Нахождение количества последовательностей символов определённой длины

Условие построения цепочки

- Один символ (A)
- Несколько символов в произвольном порядке (A, B, C)
- Несколько символов в определённом порядке (ABCABC…)
- Содержит (не содержит) определённые символы в определённом количестве или на заданных местах.

```
for i in str:
    обработать символ i
```

```
for i in range(0, len(str)):
    обработать символ i
```

Поиск максимальной цепочки, состоящей из конкретного символа

```
maxLen = 0
tekLen = 0
for i in str:
  if i == 'A':
    tekLen += 1
    if tekLen > maxLen:
      maxLen = tekLen
  else:
    tekLen = 0
```

Поиск максимальной цепочки, состоящей из одного любого символа

```
maxLen=1
tekLen=1
symbol = str[0]
for i in range(1, len(str)):
  if str[i] == str[i-1]:
    tekLen += 1
    if tekLen > maxLen:
      maxLen = tekLen
      symbol = str[i]
  else:
    tekLen = 1
```

Текстовый файл **24.txt** состоит не более чем из 10^6 символов X, Y и Z. Определите максимальное количество идущих подряд символов, среди которых каждые два соседних различны. Для выполнения этого задания следует написать программу.

```
with open( "24.txt", "r" ) as F:
  str = F.readline()
maxLen=1
tekLen = 1
for i in range(1, len(str)):
  if str[i] != str[i-1]:
    tekLen += 1
    if tekLen > maxLen:
      maxLen = tekLen
  else:
    tekLen = 1
print( maxLen )
```

В текстовом файле **24-k8.txt** находится цепочка из символов, в которую могут входить заглавные буквы латинского алфавита A…Z и десятичные цифры. Найдите длину самой длинной подцепочки, состоящей из одинаковых символов. **Для каждой цепочки максимальной длины** выведите в отдельной строке сначала символ, из которого строится эта цепочка, а затем через пробел – длину этой цепочки.

```
with open( "24-k8.txt", "r" ) as F:
  str = F.readline()
maxLen=1
tekLen=1
symbolArray = [str[0]]
for i in range(1, len(str)):
  if str[i] == str[i-1]:
    tekLen += 1
    if tekLen == maxLen:
      symbolArray.append( str[i] )
    elif tekLen > maxLen:
      maxLen = tekLen
      symbolArray = [str[i]]
  else:
    tekLen = 1
for symbol in symbolArray:
  print( symbol, maxLen )
```

В текстовом файле **24-k7.txt** находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите количество цепочек длины 3, удовлетворяющих следующим условиям:

- 1-й символ – один из символов B, C или D;
- 2-й символ – один из символов B, D, E, который не совпадает с первым;
- 3-й символ – один из символов B, C, E, который не совпадает со вторым.

```
str = open('24-k7.txt').readline()
count = 0
for i in range(len(str)-2):
    if str[i] in 'BCD' and str[i+1] in 'BDE' \
      and str[i+2] in 'BCE' and str[i]!=s[i+1] \
      and str[i+1]!=str[i+2]:
      count += 1
print(count)
```

В текстовом файле **k7.txt** находится цепочка из символов латинского алфавита A, B, C, D, E. Найдите максимальную длину цепочки вида EABEABEABE... (составленной из фрагментов EAB, **последний фрагмент может быть неполным**).

Символ	symbol	E	A	B	E	A	B	E	A
Длина цепочки	tekLen	0	1	2	3	4	5	6	7
Номер симола (0, 1, 2)	tekLen % 3	0	1	2	0	1	2	0	1

```
str = open('24-k7.txt').read()
tekLen = 0
maxLen = 0
for symbol in str:
    if (symbol == 'E' and tekLen % 3 == 0) or \
       (symbol == 'A' and tekLen % 3 == 1) or \
       (symbol == 'B' and tekLen % 3 == 2):
            tekLen += 1
            if tekLen > maxLen:
                maxLen = tekLen
            elif (symbol =='E'):
                tekLen = 1
            else:
                tekLen = 0
print(maxLen)
```

24 Текстовый файл содержит только заглавные буквы латинского алфавита (ABC…Z). Определите максимальное количество идущих подряд символов, среди которых нет ни одной буквы A и при этом не менее трёх букв E.

Ответ: _____.

Задание 25.

Обработка целых чисел.

Проверка делимости

Полный перебор

```
nDel = 0
for d in range(1, n+1):
  if n % d == 0:
    nDel += 1
if nDel == 2:
  print( "Число простое" )
else:
  print( "Число составное" )
```

Оптимизированный вариант

```
nDel = 0
for d in range(2, round(sqrt(n))+1):
  if n % d == 0:
    nDel += 1
if nDel == 2:
  print( "Число простое" )
else:
  print( "Число составное" )
```

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3532000; 3532160], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.

```
count = 0
for n in range(3532000, 3532160+1):
  countDel = 0
  for d in range(1,n+1):
    if n % d == 0:
        countDel += 1
    if countDel > 2: break
  if countDel == 2:
    count += 1
    print( count, n )
```

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [3532000; 3532160], простые числа. Выведите все найденные простые числа в порядке возрастания, слева от каждого числа выведите его номер по порядку.

```
from math import sqrt
count = 0
for n in range(3532000, 3532160+1):
  flag = True
  for d in range(2, round(sqrt(n))):
    if n % d == 0:
      flag = False
      break
  if flag:
    count += 1
    print( count, n )
```

```
import math
x=math.sqrt(n)
```

Найдите все **натуральные** числа, принадлежащие отрезку [77 777 777; 88 888 888], у которых ровно пять различных нечётных делителей (количество чётных делителей может быть любым). В ответе перечислите найденные числа, справа от каждого числа запишите его наименьший нечётный делитель, не равный 1.

$\mathbf{n = p_1^{k1} p_2^{k2} \ldots p_m^{km}}$

$\mathbf{p_i}$ – различные простые делители (i=1, ...m),

$\mathbf{k_i}$ – их кратности (натуральные числа) (i=1, ..., m) .

$18 = 2 \cdot 3^2$

Делители числа 18 – это 1 и 2, 3, $2 \cdot 3 = 6$, $3 \cdot 3 = 9$, $2 \cdot 3^2 = 18$.

Два простых нечётных числа $n = 2^k p1 p2$

n делится на 1, p1, p2 и произведение p1p2

Одно из простых чисел во второй степени: $n = 2^k p1^2 p2$

n делится 1, p1, p2, $p1^2$, p1p2, $p1^2 p2$.

Алгоритм:

1. Берём число **X** из заданного диапазона

2. Убираем из разложения числа **X** на простые множители все двойки

3. Находим корень четвёртой степени из остаточного **X**

4. Проверяем, является ли **X** четвёртой степенью простого числа

```
def isPrime( x ):
  if x <= 1: return False
  d = 2
  while d*d <= x:
    if x % d == 0:
      return False
    d += 1
  return True

start, end = 77777777, 88888888
from math import sqrt
for n in range(start, end+1):
  x = n
  while x % 2 == 0:
     x //= 2
  qX = round(sqrt(sqrt(x)))
  if isPrime(qX) and qX**4 == x:
    print( n, qX )
```

```
77 900 162    79
78 074 896    47
78 675 968    7
80 604 484    67
81 920 000    5
84 934 656    3
85 525 504    17
88 529 281    97
```

```
start, end = 77777777, 88888888
primes = [2]
for i in range(3, int(end**0.25) + 1, 2):
    for d in range(2, int(i**0.5) + 1):
        if i % d == 0:
            break
    else:
        primes.append(i)

ans = []
for el in primes[1:]:
    num = el**4
    while num <= end:
        if num >= start:
            ans.append([num, el])
        num *= 2
print(*sorted(ans), sep='\n')
```

Напишите программу, которая ищет среди целых чисел, принадлежащих числовому отрезку [174457; 174505], числа, имеющие ровно два различных натуральных делителя, не считая единицы и самого числа. Для каждого найденного числа запишите эти два делителя в таблицу на экране с новой строки в порядке возрастания произведения этих двух делителей. Делители в строке таблицы также должны следовать в порядке возрастания.

```
from math import sqrt
divCount = 2
for n in range(174457, 174505+1):
  divs = []
  q = int(sqrt(n))
  for d in range(2,q+1):
    if n % d == 0:
      if d == n//d:
        divs = divs + [d]
      else:
        divs = divs + [d, n//d]
      if len(divs) > divCount: break
  if len(divs) == divCount:
    print( *divs )
```