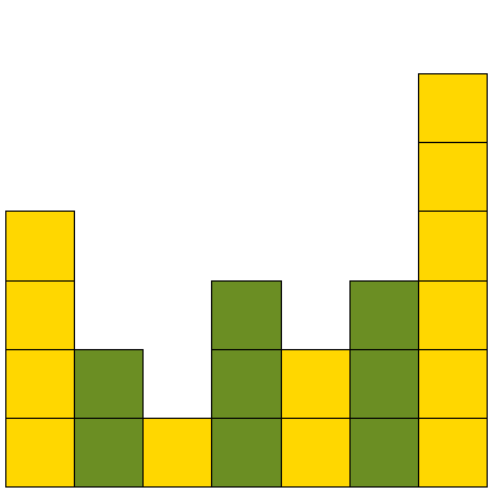
Zadanie Wieże – LOGIA 22 (2021/22), etap 2

Treść zadania

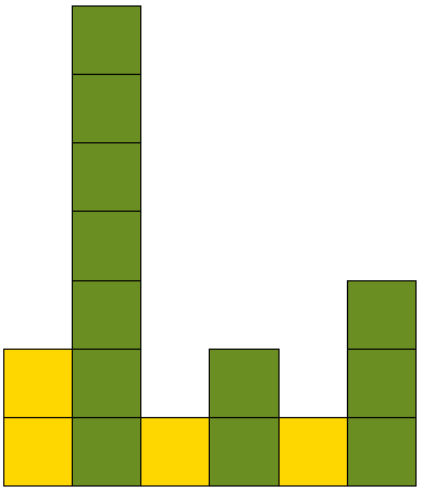
Bartek rysuje wieże, które opisuje za pomocą liter **G** i **Y**. Każda wieża składa się z zamalowanych kwadratów. Liczba kolejnych takich samych liter określa wysokość wieży, a kolor oznaczony jest przez literę: Y – żółty i G – zielony.

Pomóż Bartkowi i napisz funkcję **wie(opis)** rysującą ciąg wież według opisanej zasady. Parametr **opis** to napis złożony z liter **G** i **Y** o długości co najwyżej **52**, składający się z co najmniej jednej litery **G** i co najmniej jednej litery **Y**. Rysunek powinien być jednakowo oddalony od lewej i prawej krawędzi ekranu oraz mieć szerokość lub wysokość **400**. Drugi wymiar nie może być większy niż **400**.

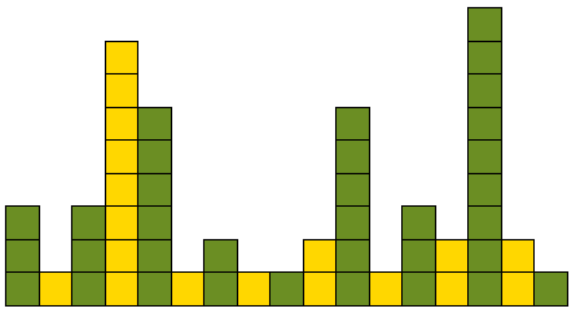
Przykłady:



wie('YYYYGGYGGGYYGGGYYYYYY')



wie('YYGGGGGGGGYGGYGGG')



wie('GGGYGGGYYYYYYYYGGGGGGYGGYGYYGGGGGGGYGGGYYGGGGGGGGGGYYG')

Omówienie rozwiązania

Zanim zaczniemy rysować wieże należy obliczyć długość boku kwadratu dla danego parametru. W treści zadania jest powiedziane, że szerokość lub wysokość rysunku musi wynosić 400. Drugi wymiar nie może być większy niż 400. Liczba kwadratów w poziomie jest zależna od liczby podciągów złożonych

z takich samych liter **G** lub **Y**. Natomiast największa wysokość wieży (liczba kwadratów wieży) jest zależna od maksymalnej długości podciągu takich samych liter **G** lub **Y**.

Dla pierwszego przykładu mamy siedem wież, a najwyższa wieża składa się z sześciu kwadratów. Drugi przykład to sześć wież, najwyższa składa się z siedmiu kwadratów. Obliczając długość boku kwadratu musimy 400 podzielić przez większą z tych dwóch wartości – liczby wież i wysokości.

W tym celu definiujemy funkcję **mx(opis)**, której wynikiem jej jest liczba rysowanych wież. Wynikiem drugiej funkcji – **my(opis)** – jest długość maksymalnego podciągu takich samych liter **G** lub **Y**, czyli wysokość najwyższej wieży.

```
def mx(opis):
    x = 1
    for i in range(len(opis) - 1):
        if opis[i] != opis[i + 1]:
            x += 1
    return x

def my(opis):
    y = 1
    maks = 1
    for i in range(len(opis)-1):
        if opis[i] == opis[i + 1]:
            y += 1
        else:
            if maks < y:
                maks = y
            y = 1
    if maks < y:
                maks = y
    return maks
```

W funkcji głównej zaczynamy od obliczeń, wynikiem funkcja **max()** jest większa wartość z podanych jako parametry. W ostatniej linii znajduje się kontrolne wypisywanie szerokości i wysokości rysunku, aby sprawdzić poprawność obliczeń. Przetestujmy ten fragment dla różnych wartości parametru `opis`. Jedna z wypisanych wartości musi wynosić 400, druga musi być mniejsza.

```
def wie(opis):
    szer = mx(opis)
    wys = my(opis)
    bok = 400 / max(szer, wys)
    print(szer*bok, wys*bok)
```

Jeśli obliczenia są prawidłowe, to należy wstawić znak komentarza # przed instrukcją `print()`, oprócz wynikowego rysunku nie powinny być wypisywane żadne komunikaty.

Na początku przenosimy żółwia w takie miejsce, aby narysowany rysunek był jednakowo odległy od lewej i prawej strony ekranu. Warto też zacząć rysowanie niżej (nie ma wymogu środkowania w pionie), aby dla wszystkich parametrów rysunek był dobrze widoczny.

Warto też zdefiniować pomocniczą funkcję rysowania kwadratu, która oprócz parametru `bok` może mieć drugi parametr – kolor zamalowania.

```
def kwad(bok, kolor):
    fillcolor(kolor)
    begin_fill()
    for i in range(4):
        fd(bok); lt(90)
    end_fill()
```

Tworzenie wynikowego rysunku polega na przeglądaniu parametru `opis` znak po znaku i rysowaniu kolejnych kwadratów. Jednakowe kolorystycznie kwadraty są rysowane jeden nad drugim, po zmianie koloru żółw musi wrócić na poziom, w którym zaczynał rysowanie poprzedniej wieży i zacząć rysować kolejną.

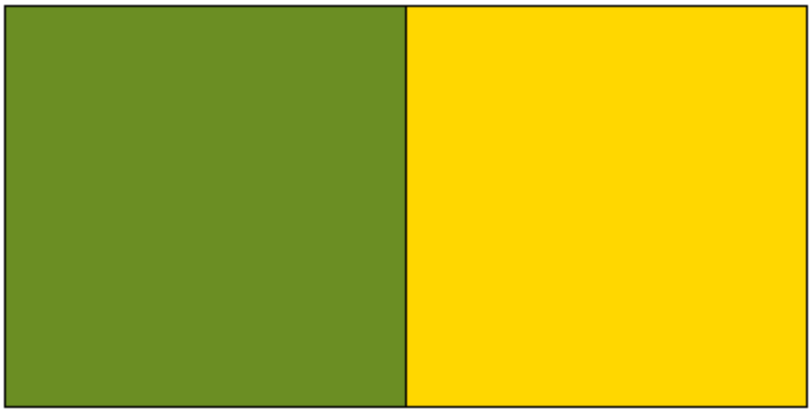
Rozwiązanie w języku Python

```
from turtle import *

def kwad(bok, kolor):
    fillcolor(kolor)
    begin_fill()
    for i in range(4):
        fd(bok); lt(90)
    end_fill()

def mx(opis):
    x = 1
    for i in range(len(opis) - 1):
        if opis[i] != opis[i + 1]:
            x += 1
    return x

def my(opis):
    y = 1
    maks = 1
    for i in range(len(opis)-1):
        if opis[i] == opis[i + 1]:
            y += 1
        else:
            if maks < y:
                maks = y
            y = 1
    if maks < y:
                maks = y
    return maks

def wie(opis):
    szer = mx(opis)
    wys = my(opis)
    bok = 400 / max(szer, wys)
    #print(szer*bok,wys*bok)
    pu(); bk(szer * bok / 2);lt(90);bk(200);rt(90);pd()
```

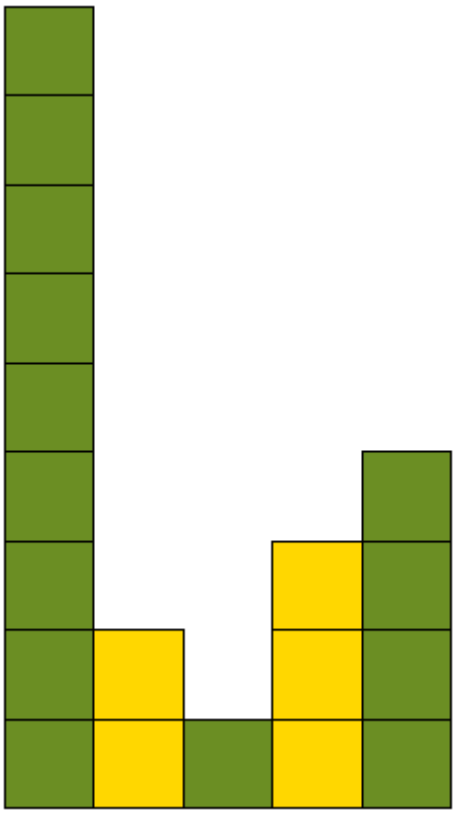
```
37      for i in range(len(opis)):
38          if opis[i] == 'Y':
39              kwad(bok,"gold")
40          else:
41              kwad(bok,"olivedrab")
42          if i != len(opis)-1 and opis[i] == opis[i+1]:
43              lt(90);fd(bok);rt(90)
44          else:
45              sety(-200)
46              fd(bok)
```

Testy

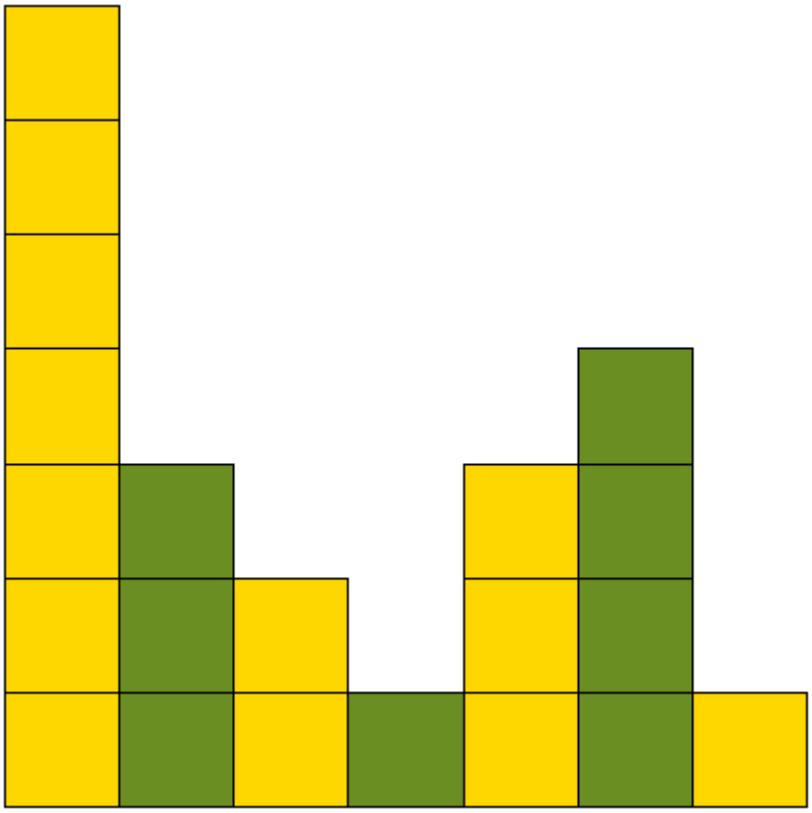
Wywołujemy funkcję **wie()** dla różnych parametrów, tak żeby sprawdzić skalowanie rysunku zarówno w pionie jak i poziomie oraz wartości graniczne.



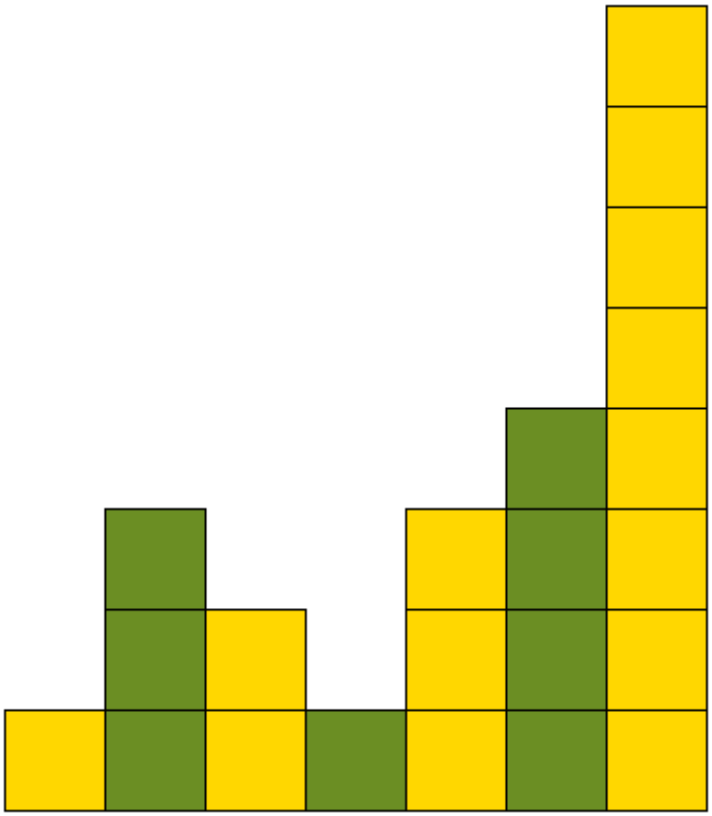
wie('GY')



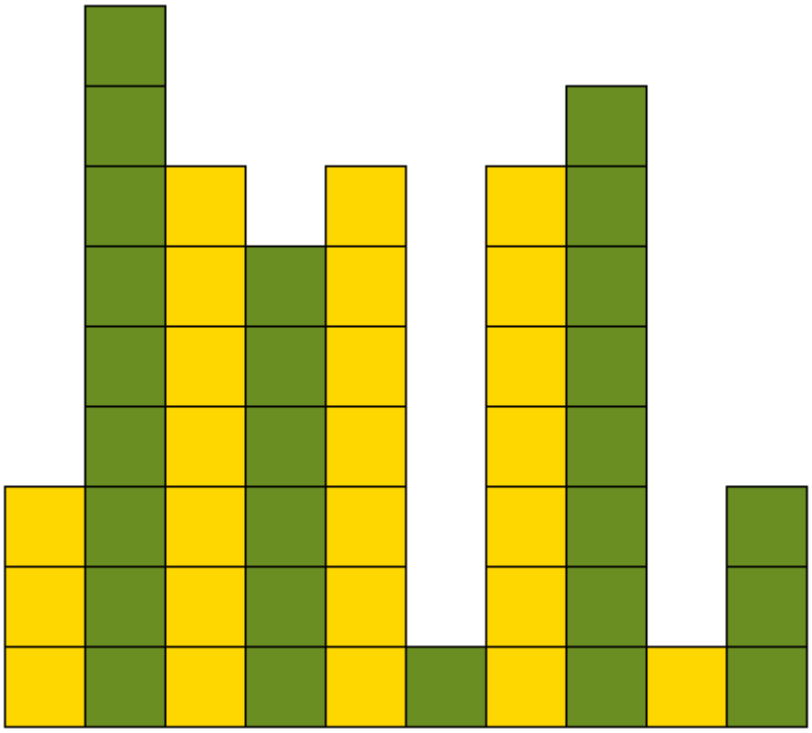
wie('GGGGGGGGGYYGYYYGGGG')



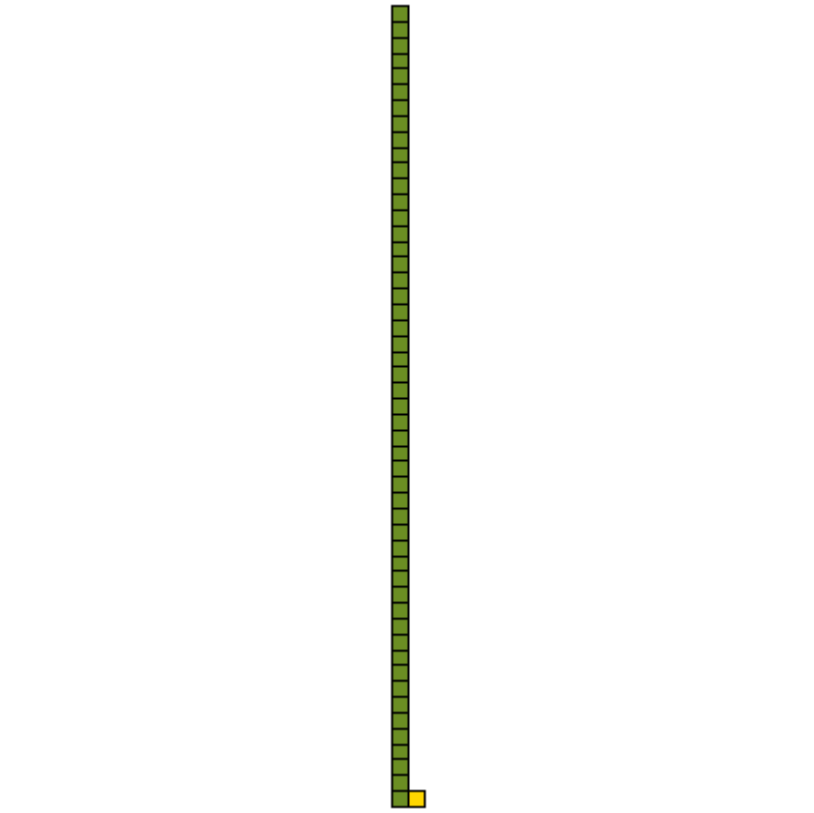
wie('YYYYYYYGGGYYGYYYGGGGY')



wie('YGGGYYGYYYGGGGYYYYYYYY')



wie('YYYGGGGGGGGGYYYYYYYGGGGGGYYYYYYYGYYYYYYYGGGGGGGGGYGGG')



wie('G'*51+'Y')

wie('YG'*26)

Zadanie Nasłuch – LOGIA 22 (2021/22), etap 2

Treść zadania

Sławek pracuje w Centrum Astronomicznym Bitmir. Jego zadaniem jest sprawdzanie zgodności wzorów w sygnałach radiowych. Każdy sygnał reprezentowany jest przez ciąg liter. W żargonie trafienie „wow!” następuje wtedy, gdy w dwóch porównywanych sygnałach litera na tej samej pozycji okaże się taka sama. Jeżeli litera nie występuje na tej samej pozycji, ale występuje w obu sygnałach, to mamy trafienie „ooo!”. Wystąpienie litery może być wykorzystane tylko w jednym trafieniu, pierwszeństwo ma trafienie „wow!”. Na przykład w sygnałach "aaaaabbb" i "addcccaa" występuje jedno trafienie „wow!” i dwa trafienia „ooo!”.

Napisz program, który wczyta ze standardowego wejścia dwa sygnały i wypisze wynik porównania zgodności sygnałów w postaci liczby całkowitej nieujemnej, której cyfra dziesiątek oznacza liczbę trafień „wow!”, a cyfra jedności liczbę trafień „ooo!”.

Wejście:
Dwa napisy złożone z małych liter alfabetu łacińskiego oddzielone spacją o długości od 1 do 9 każdy.

Wyjście:
Liczba całkowita z przedziału [0,90], której cyfra dziesiątek oznacza liczbę trafień „wow!”, a cyfra jedności liczbę trafień „ooo!”.

	Przykład 1	*Przykład 2*	*Przykład 3*
Wejście	**aaabbb bbbaaabbb**	**abcc aacawxyz**	**aaabbb aabb**
Wyjście	**6**	**20**	**31**
Komentarz	Powtarzają się trzykrotnie litery „a” oraz „b”. Żadna litera nie występuje na tej samej pozycji.	Litery „a” oraz „c” znajdują się na tych samych pozycjach i tworzą trafienia „wow!”.	Dwie litery „a” i litera „b” tworzą trafienia „wow!”. Jedna litera „b” tworzy trafienie „ooo!”.

Omówienie rozwiązania

Zadanie jest uogólnieniem zadania „Nasłuch” z pierwszego etapu Konkursu Logia w roku szkolnym 2021/22. Porównywane słowa mogą być różnej długości, litery mogą się powtarzać. Techniczną zmianą jest zastąpienie 10-ciu cyfr przez 26 liter.

W przykładowym rozwiązaniu wykorzystamy pomocniczą tablicę, w której będziemy zaznaczać wystąpienia poszczególnych liter. Litery, które występują na tych samych pozycjach zwiększą wartość zmiennej trafienia o 10, ponadto litery z napisu U będą zwiększały o 1 odpowiadający im numer w tablicy zliczone, zaś litery z napisu W będą zmniejszały o 1 odpowiadający im numer (litera „a” ma numer 0, „b” - 1,…, „z” - 25). Zsumowanie wartości bezwzględnych w tej tablicy da liczbę niesparowanych liter. Odejmując tę wartość od sumy długości napisów i dzieląc przez dwa otrzymamy liczbę par w obu napisach. Należy odjąć pary z trafieniami „wow”, aby otrzymać liczbę trafień „ooo”.

Rozwiązanie w postaci listy kroków:

```
wczytaj U, W
trafienia ← 0
zainicjuj tablicę zliczone[0..25] zerami
maks_dlugosc ← maksimum(dlugosc(U), dlugosc(W))
dla każdego i z [0, 1, 2,…, maks_dlugosc - 1] wykonuj kroki 6-12
    jeżeli i < dlugosc(U) wykonaj kroki 7-11
    w przeciwnym razie wykonaj krok 12
        zwiększ o 1 wartość zliczone[numer i-tego znaku z U]
        jeżeli i < dlugosc(W) wykonaj kroki 9-11
            zmniejsz o 1 wartość zliczone[numer i-tego znaku z W]
            jeżeli znaki o numerach i są równe w U i W wykonaj
                krok 11
                zwiększ o 10 wartość trafienia
    zmniejsz o 1 wartość zliczone[numer i-tego znaku z W]
suma ← 0
dla każdego j z [0, 1, 2,…, 25] wykonaj krok 15
    suma ← suma + wartość_bezwzględna(zliczone[j])
trafienia ← trafienia + (dlugosc(U) + dlugosc(W) - suma) div 2
            - trafienia div 10
wyświetl trafienia
```

W pierwszych trzech liniach wczytujemy dwa napisy oraz ustawiamy początkowe wartości zmiennych na zero. W linii 4 obliczamy długość dłuższego napisu i zapisujemy w zmiennej - tyle razy będziemy iterować w liniach 6-12. W liniach tych zwiększana jest wartość zmiennej `trafienia`, gdy litery na takich samych pozycjach są równe. Ponadto w tablicy zmieniane są wartości wystąpień odpowiadające numerom liter. W liniach 13-15 obliczana jest suma wartości bezwzględnych z tabeli `zliczone`. Następnie modyfikowana jest zmienna trafienia, w której mieliśmy zliczone trafienia „wow”, a teraz dodajemy trafienia „ooo”.

Rozwiązanie w języku Python

Wykorzystując rozwiązanie w postaci listy kroków napisanie rozwiązania w Python mogłoby wyglądać następująco:

```
 1 U, W = input(), split()
 2 trafienia = 0
 3 zliczone = 26*[0]
 4 maks_dlugosc = max(len(U), len(W))
 5 for i in range(maks_dlugosc):
 6     if i < len(U):
 7         zliczone[ord(U[i]) - ord('a')] += 1
 8         if i < len(W):
 9             zliczone[ord(W[i]) - ord('a')] -= 1
10             if U[i] == W[i]:
11                 trafienia += 10
12     else:
13         zliczone[ord(W[i]) - ord('a')] -= 1
14 suma = 0
15 for j in range(26):
16     suma += abs(zliczone[j])
17 trafienia += (len(U) + len(W) - suma) // 2 - trafienia // 10
18 print(trafienia)
```

Do obliczenia numeru litery wykorzystaliśmy funkcję `ord(z)`, której wartością jest kod ASCII znaku `z`. Odejmując kod ASCII litery „a” otrzymamy numer litery.

Inne rozwiązanie, w którym wykorzystujemy operacje na listach składanych:

```
1  U, W = input().split()
2  trafienia = 9 * sum(1 for i in range(min(len(U),len(W))) if U[i]==W[i])
3  trafienia += sum(min(U.count(z), W.count(z)) for z in set(U) & set(W))
4  print(trafienia)
```

W linii 2 zliczamy trafienia „wow” lecz nie mnożymy ich przez 10, lecz 9, aby w linii 3 już ich nie odejmować, jak to zrobiliśmy w poprzednim programie w linii 17. Sumujemy funkcją `sum()` listę złożoną z tylu jedynek, ile liter jest równych w obu napisach. W linii 3 lista składana jest tworzona przez zliczanie liczby liter w obu napisach i wybrania wartości mniejszej do sumy. Należy zauważyć, że zliczamy w ten sposób także trafienia „wow”. Bierzemy pod uwagę tylko te znaki, które występują w obu napisach. Konstrukcja `set(U) & set(W)` zwraca część wspólną obu zbiorów, czyli litery, które występują w obu napisach `U` oraz `W`. Z kolei metoda `U.count(z)` pozwala na zliczenie liczby wystąpień znaku z w napisie `U`.

Testy

Testy zostały tak dobrane, by sprawdzić m.in. czy są zliczane litery występujące na początku lub końcu napisu, jak zachowuje się program, gdy litery powtarzają się.

Grupa testów	Test	Wynik
I	abc cba	12
	abcd uxyz	0
	abcde bxyze	11
II	fghj jf	2
	kmnoprs skmnoprs	7
	abzaedfa zaaaaaada	14
III	zzzzzzza azzzzzzzz	72
	bbaaaaaaa aabb	4
	ddddddddd eddddddddd	80
IV	cbabc abcba	22
	kmn kmnkmnkmn	30
	abcdedcba edcbabcde	26
V	fg fhgjfgfgf	11
	rtyeertye yrerter	6
	wer twerwerte	3

Zadanie Liczby Janka – LOGIA 22 (2021/22), etap 2

Treść zadania

Janek wybiera liczby złożone (większe od 1 niebędące liczbami pierwszymi), dla których średnia arytmetyczna wszystkich dzielników właściwych (mniejszych od liczby) nie jest większa od pierwiastka kwadratowego z tej liczby. Na przykład dzielniki właściwe liczby 45 to 1, 3, 5, 9, 15, średnia arytmetyczna wynosi 6,6, a pierwiastek kwadratowy jest większy niż 6,7.

Pomóż Jankowi i napisz program, który wczyta liczbę i wypisze 5 liczb większych od podanej liczby spełniających opisane warunki. Liczby powinny być jak najmniejsze i wypisane w kolejności rosnącej. Postaraj się, żeby na wynik nie trzeba było czekać zbyt długo.

Wejście:

Liczba naturalna z zakresu od 2 do 100 000 000.

Wyjście:

Pięć liczb naturalnych oddzielonych spacją spełniających opisane warunki.

	Przykład 1	*Przykład 2*	*Przykład 3*
Wejście	**40**	**302**	**1200**
Wyjście	**45 49 51 55 65**	**319 323 341 361 377**	**1207 1219 1241 1247 1271**

Omówienie rozwiązania

Zadanie polega na wczytaniu liczby naturalnej z podanego zakresu i wypisaniu pięciu kolejnych liczb większych od podanej spełniających warunki zadania. Będziemy sprawdzać kolejne liczby w pętli począwszy od wczytanej liczby powiększonej o 1, aż nie znajdziemy pięciu liczb Janka. Do sprawdzania będzie przydatna pomocnicza funkcja, w której będziemy badać dla danej liczby, czy średnia arytmetyczna dzielników właściwych jest większa od pierwiastka kwadratowego tej liczby. Zapiszmy podaną zależność wzorem, gdzie d_1, d_2, … , d_n to dzielniki właściwe liczby, a n – liczba tych dzielników.

$$\frac{d_1 + d_2 + \ldots + d_n}{n} > \sqrt{liczba}$$

Ponieważ w komputerze liczby rzeczywiste są pamiętane z pewnym przybliżeniem, lepiej używać wyłącznie liczb całkowitych. Także działania na liczbach całkowitych są wykonywane przez komputer szybciej niż na liczbach rzeczywistych. Dlatego przekształcimy naszą nierówność mnożąc obie strony przez *n*, a następnie podnosząc obie strony do kwadratu. Warto zauważyć, że mamy do czynienia z liczbami dodatnimi, więc podnoszenie do kwadratu nie powoduje błędu.

$$d_1 + d_2 + \ldots + d_n > n\sqrt{liczba}$$

$$(d_1 + d_2 + \ldots + d_n)^2 > n^2\, liczba$$

W tej postaci możemy sprawdzić, czy zachodzi nierówność, używając wyłącznie działań na liczbach całkowitych. Pozostaje zaimplementować rozwiązanie.

Rozwiązanie w języku Python

Właściwe rozwiązanie zadania rozpoczyna się od linii 18, wcześniej jest zdefiniowana pomocnicza funkcja. Funkcja `liczba_janka(liczba)` ma jeden parametr – liczbę i sprawdza, czy dla niej zachodzi opisana nierówność. Wynikiem funkcji jest prawda (`True`) lub fałsz (`False`). Zapis `pom = [1]` oznacza utworzenie listy złożonej z jednego elementu 1, a zapis `pom = pom + [i]` dodanie do listy pom elementu `i`. Warto jeszcze zwrócić uwagę, że w rozwiązaniu zastosowano funkcję `print()` z parametrem end, który przedefiniowuje standardowe zachowanie funkcji `print`. Domyślnie funkcja wypisuje na ekranie wartość i powoduje przejście kursora na następnej linii. Dodanie parametru `end = " "` powoduje, że zamiast przejścia do nowej linii będzie wypisana spacja.

```
 1  def liczba_janka(liczba):
 2      pom = [1]
 3      i = 2
 4      while i * i < liczba:
 5          if liczba % i == 0:
 6              pom = pom + [i] + [liczba // i]
 7          i += 1
 8      if i * i == liczba:
 9          pom = pom + [i]
10      suma = sum(pom)
11      if len(pom) < 2:
12          return False
13      else:
14          return suma * suma <= len(pom) * len(pom) * liczba
15
16  liczba = int(input())
17  ile = 0
18  while ile < 5:
19      liczba += 1
20      if liczba_janka(liczba):
21          ile += 1
22          print(liczba, end = " ")
```

Warto zwrócić uwagę na zakres pętli `while` budującej listę dzielników właściwych (linia 4). Wystarczy sprawdzać potencjalne dzielniki liczby do pierwiastka z niej i dodawać do listy dzielników od razu dwa dzielniki. Ma to bardzo duży wpływ na czas działania programu. Po pętli sprawdzamy dodatkowo (linia 8), czy liczba nie jest kwadratem, wówczas dodajemy do listy jeszcze pierwiastek z liczby.

Testy

Wywołujemy program dla różnych testów tak, aby daną była zarówno liczba mała, jak i duża. Rozwiązanie warto testować dla różnych przypadków, gdyż przy implementacji z użyciem liczb rzeczywistych (zmiennoprzecinkowych) mogą wystąpić błędy zaokrągleń.

Test	Wynik
46	49 51 55 65 77
134	143 145 155 161 169
235	247 253 259 287 289

77	85 91 95 115 119
456	473 481 493 517 527
567	583 589 611 629 649
787	793 799 803 817 841
9672	9701 9727 9761 9797 9799
12345	12349 12361 12367 12371 12403
1236577	1236673 1236679 1236731 1236793 1236863
4565633	4565699 4565713 4565767 4565779 4565867
98989898	98989909 98989927 98989987 98990029 98990179
10000001	10000043 10000153 10000163 10000183 10000217
10000534	10000547 10000567 10000649 10000703 10000751
100000000	100000139 100000169 100000183 100000253 100000423

Zadanie Szyfr kolumnowy – LOGIA 22 (2021/22), etap 2

Treść zadania

Alicja do szyfrowania wykorzystuje klucz złożony z liter. Klucz przekształca na ciąg niepowtarzających się liczb naturalnych o wartościach od 1 do długości klucza. Literom klucza przypisuje liczby zgodnie z kolejnością alfabetyczną liter klucza. Jeżeli w kluczu występuje kilka takich samych liter, to każde kolejne wystąpienie tej litery otrzymuje liczbę o jeden większą od liczby przypisanej poprzedniemu wystąpieniu. Na przykład klucz "oczko" Alicja przekształca na ciąg 3 1 5 2 4.

Szyfrowaną wiadomość Alicja wpisuje do tabeli o szerokości równej długości klucza. Tekst wiadomości wpisuje wierszami, ostatni wiersz może być niepełny. Zaszyfrowaną wiadomość otrzymuje odczytując tabelę kolumnami według kolejności określonej przez ciąg liczbowy wyznaczony na podstawie klucza. Na przykład dla wiadomości "konkurslogia" i klucza "oczko" otrzymuje napis "nlkriugosako".

1	2	3	4	5
k	o	n	k	u
r	s	l	o	g
i	a			

Pomóż Alicji i napisz program, który wczyta wiadomość i klucz, a następnie wypisze zaszyfrowaną wiadomość.

Wejście:

Dwa niepuste napisy oddzielone spacją złożone z małych liter alfabetu łacińskiego o długości co najwyżej 1000 liter każdy.

Wyjście:

Zaszyfrowany napis.

	Przykład 1	*Przykład 2*	*Przykład 3*
Wejście	**konkurslogia oczko**	**tajnawiadomosc plot**	**spotkanieodwolane kajak**
Wyjście	**nlkriugosako**	**jimtadsawocnao**	**telsadnoiopnwekoa**

Omówienie rozwiązania

W zadaniu wczytujemy dwa napisy, z których pierwszy zawiera napis do zaszyfrowania, a z drugiego budujemy klucz. Żeby zbudować klucz należy posortować podane litery, by wiedzieć w jakiej kolejności odczytywać kolumny. Trzeba też zadbać, by takie same litery brać w kolejności występowania. Ten problem można rozwiązać znajdując pierwsze wystąpienie litery i zastąpienie jej znakiem, który nie występuje w kluczu, np. „*”.

Dla danej „oczko” posortowane litery to: c, k, o, o, z. Zapisanie liter w listy pomoże nam wyznaczyć kolejność kolumn. Wystarczy analizować kolejne litery podanego klucza i znajdować dla każdej z nich indeks w posortowanej liście. Jednocześnie wyszukany znak zastępujemy gwiazdką.

Teraz możemy przystąpić do szyfrowania – wypisywania szyfrowanego napisu kolumnami według wyznaczonej kolejności. Dla słowa „konkurslogia” będzie to wyglądać następująco:

0	1	2	3	4
k	o	n	k	u
r	s	l	o	g
i	a			

2 – nl, 0 – kri, 4 – ug, 1 – osa, 3 – ko, co daje szyfrogram „nlkriugosako”. Warto zauważyć, że indeksowanie kolumn od 0 jest łatwiejsze w implementacji niż indeksowanie od 1.

Rozwiązanie w języku Python

Pierwsza linie kodu to wczytanie danych. Słowo – klucz jest zamieniany na listę, a następnie lista jest sortowana funkcją `sorted()`. Do znajdowania pozycji litery na liście wykorzystujemy metodę `index()`.

Po zbudowaniu klucza – wygenerowaniu kolejności kolumn, przystępujemy do właściwego szyfrowania. W języku Python można budować nowe napisy biorąc niespójny fragment oryginalnego napisu. Wystarczy skorzystać z funkcji `napis[od:do:krok]`. W przypadku tego zadania odpowiednią kolumnę wyodrębniono za pomocą wyrażenia `w[x::dl]`. W wyrażeniu drugi parametr został pominięty, gdyż przyjmuje domyślną wartość równą długości napisu `w`.

```
 1  w,klucz = input().split()
 2
 3  #budowanie klucza
 4  klucz_s = sorted(list(klucz))
 5  kolumny = []
 6  for x in klucz:
 7      i = klucz_s.index(x)
 8      kolumny += [i]
 9      klucz_s[i] = '*'
10
11  #szyfrowanie
12  dl = len(klucz)
13  for i in range(dl):
14      x = kolumny[i]
15      print(w[x::dl], end = "")
```

Testy

Wywołujemy program dla różnych testów. Warto w pierwszym etapie uwzględnić testy krótsze, które można przeanalizować ręcznie, a potem przejść do dłuższych. Prostsze przypadki obejmują przykłady, w których w kluczu nie pojawia się żadna litera więcej niż raz.

Test	Wynik
spotkaniewpiatek gfed	tiikonpepawtskea
akcjaodwolana akbjdnfm	aooklaacawjnd
jutrooczwartej oifgklmpks	ztejrutrjocwoa
dziwnytelefon kajak	wldyfienztone
zostajemywdomu mama	sedzaymtmoojwu
osiodmejrano babababa	dormsaeinjoo
samochod alalalalalala	sdamocho
brakdanych comipowiesz	badaynhkrc
przegrana falszywawiadomosc	rpnrazgae
abcdefghijklmnopqrstuvwxyz dcbdcbdcb	gpydmvajshqzenwbktirfoxclu
zyxwvutsrqponmlkjihgfedcba zyxwvutsrqponmlkjihgfedcba	abcdefghijklmnopqrstuvwxyz

Test

mjroqfdtvpgcyawkzsxnhilbeuzbatviedumfowsnkljrqgphxcyfweiolxhumjbcsrdtpagyvnzkqcdfubwsvrqij
metaypznxhgoklkzhatrfqjysepxidulobnmgwcvfzqoglwajpvtbyruxmsnkedchigdlpvwcobtzakmeqfrsihxu
nyjgryvpsbitfcdelajnuzqwhmkoxpyqgxnjseabhktldovwumczrifcyuamwlhtpjkrqxogsefvbdinzliyurezgvn
akobmhfcdxspqtjwohrjbsexfumdzypiltawkcqgvntrmwdysqufcajpxikeohnlvbzgayfnmwjqxpibtkdoslzec
urhvgqwcjdrsokgftzvmuinypxealbhclkhabdxyqmefnrpztsouvjiwgajmkrwdvyepoqcfgxibszunthlksyoxrw
gfdpmlqbjntiaezuvchdhgnwapyrzqxctbvilekmusojfswtoecxyhzguijnpavqrlmdbkfinmtsopjxuagbedrqkc
fvzhlywtgbdwuyinjhpeaormlfcqxkzvsxupghlrcvfqzdyoknbeitmsjawvdwixuqlhonecbkmsgpyrzftajowazf
gucpqdieytjkxmvhbrlnsbqfvposwclhrxiujkedantgzymajdpkcnfebhyqrvimtosglwuzxfjohryzuabdpnwigql
tscxvmekemokzwncvsuxibpjflrgdytqahjzsqpebhlwofigrtuaymknvxcddjvtcgamyerzhnflwxkpiusbqoixlyr
thbfowjzuqksdampevncgonrihyesflqkxcampbzutdjgvwzpywgxbsefvjlidorhtqckmnauejfmqpkhdvbwtzl
sirunyoxgcaiedhpsywjufrctbkgqaxnzvomlyvqoxbsakwnhimcedjrgflzutpbmnqgrdwevcitfauksojzyxphlzx
dfbmotagwv stuuts

mdyxevflheurycsmzktsugqjrkgcksypcnmqelmclrenrafqrfpktsjozmisrckmxcdfswrpxnyfbedpcejegadmx
dvtyefvhqnswhkrouemnphkgdevgfzntezuftslrkdahkqrwsvrfatzbltaqbixdjbnysirtgckxdarvwhzewrcojtnf
rtxkfpocgvxdlbehgbezoxbozutxvlzodjbmlqmuxnajtzvdfiakyruadqbhxpnugrbmsxiqksaqhbnoqxrdeaxns
fapthbsxdykhmwoaicenirapouvvyfiihzachqpjyedcektucpfgvqkcfbdtohbwqgzlaungfxcakbiyghjinfwbshv
zfuytawpjkwimjgdyvmsoezavdckvfxdcqszybamzjmyfzkwlicwqisthnlipbcywhmkpvotvqcpafdkrfcunanq
ozrdqvovpdjbiyccrwslfqpoexzvgvrmfdlyiycamxndzneazzofcsbtmkpwhsgqwjpoaydmzayniwarnvfjhystu
fwksiunhphxywnyaebnpouwovphbetikegusgqahaxloozpmnjefwuplzgfkmdlbyjgixlvljtjfrsxyplmksjyzhg
mdgzxbyokenscuwxjhnmvsoewtxlbhjuqvuyxtopkibdsqylbpzuqahzqxbvltmcptqxrilqxnhvrapobigbxwsd
tgwfilyqkegrtnlhqpvjvcslrmtvnzvuwyjrfogkldjrxucyiwuegtzqjbqwiamcytuhbgxmcbghefaxteluxbumgyk
bgwfokjhznasrqoowelmwfjlmvjtanuiojximdvdvenlrelwopuedomijsdukgadcyhqfzekctjuicrqphgwblhjgu
elxnogwoitodjzhytkfcuvbturzgiacslzbjiomzwcyvsorezpbiljuwskwxlqqwtnjmnpotjdnilmdpsiqupwrghuk
zvamgprispfg

Test

abcdefghijklmnopqrstuvwxyz
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab

abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
abababababababababababababababababababababababababababababababababababababababababababab
ababababababab

abcdefghijklmnopqrstuvwxyz

Test

deizajberyqsahmvxwkftznlidojpcugosmnrutevxhjaqzkgyipdfbcwlqtashnfvzkeyxpijbmolrdugcwhpmzrn
xjfbgdvaecliytqskuwowpbeyuinvolzchdfmrxqgkjastylgropskfxjubztmihewaqncvdihgwkxduqpnysjozbv
mtlacerfncupfogxdkzrlyaqmwvbijhtesrievwfktbjpomxnyzhlgaqsduccigqfyxrhobndvwakmljszupetirznlt
oasmjpqxwgdfbkchvuyejfsbchzaoxmqkdnyewtlruigpvxrnyjhiatlmgfcpbwuvdeqo jkasdscdsed

bwpeisiwgsirgzvplomimdhjlgjqiavekcvphjhdknxrtdnagwexuostdfkgtddsnsaczlrcwzjkewofzjkhgvemche
rfrfuxdnbpvuvqomiycxvvncbzofsdpleihinzledxibhsxaxbcltbgfaipvatypytghffmmawogpihsedbwycnuab
bnvmqealmqwkrnlpcafwkznrhtlqwtjhzwnczmdrkqyujyedtjqdvuyejaykrquoljbqzojbvozeolkshgjrkifzidp
skcyxgojxjtyapcbqumlbcqtfqrosrlnwemuiuavougtxgftyfyunumpaspqxmzxyxrhw

Test

qwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghj
klzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuio
pasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnq
wertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjkl
zxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopa
sdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwer
tyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcv
bnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfg
hjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyui
opasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnq
wertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbnqwertyuiopasdfghjklzxcvbn ugfdaddddsaa

sbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxpcoxizulyk
tjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizuloxizulyktjrhegwfqdns
bavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulykrhegwfqdnsbavpcoxizulyktjrheg
wfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnqdnsbavpcoxizulyktjrhegwfqdnsbavpcox
izulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpctjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxi
zulyktjrhegwfqdnsbavpcoxizulyktjrhegwfyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrh
egwfqdnsbavpcoxizulyktjrhegulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdns
bavpcoxizulyktjrhizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxiz
ulyktjavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizwfqd
nsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavegwfqdnsbav
pcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsbavpcoxizulyktjrhegwfqdnsb