Zadanie Gwiazdy – LOGIA 19 SP (2018/19), etap 2

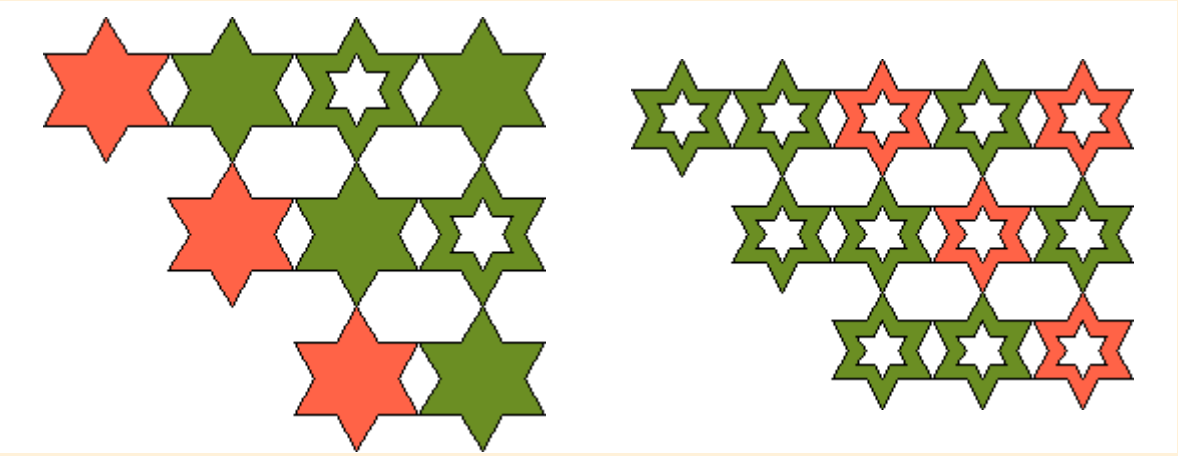
Treść zadania

Firma produkująca ozdoby zakupiła maszynę sterowaną za pomocą kodów liczbowych. Maszyna produkuje ozdoby ułożone w trzech wierszach. Pierwszy wiersz zawiera elementy ozdoby (gwiazdki) odpowiadające kolejnym cyfrom kodu liczbowego, drugi odpowiada kodowi bez ostatniej cyfry, trzeci – kodowi bez dwóch ostatnich cyfr. Maszyna wykorzystuje cztery rodzaje gwiazdek odpowiadające cyfrom. Długość boku wewnętrznej gwiazdki jest równa połowie długości boku zewnętrznej.

0 1 2 3 4 5 6 7 8 9

Napisz jednoparametrową procedurę/funkcję **gwiazdy**, po wywołaniu której na środku ekranu powstanie rysunek układu dwukolorowych gwiazdek. Parametrem jest liczba z zakresu od 1000 do 1999999999. Szerokość rysunku wynosi 560.

Przykłady:



gwiazdy(2490) **gwiazdy(11357)**

Omówienie rozwiązania

Zadanie polega na rysowaniu układów złożonych z czterech rodzajów gwiazdek: czerwonych i zielonych, z wycięciem w środku lub bez. Gwiazdki rysowane są w trzech rządkach wyrównanych do prawej, przy czym każdy kolejny rządek jest o jedną gwiazdkę krótszy. Rodzaj kolejnych gwiazdek w pierwszym wierszu układu został opisany za pomocą kodu liczbowego – każdej cyfrze przyporządkowana została pewna gwiazdka zgodnie z systemem zdefiniowanym w treści zadania. Możemy zauważyć, że układ gwiazdek odpowiadających kolejnym cyfrom powtarza się cyklicznie co 4 pozycje – można to wykorzystać pisząc funkcję rysującą kolejne gwiazdki, typ gwiazdki określimy za pomocą operacji modulo (reszta z dzielenia przez 4). Warto przygotować funkcje pomocnicze: **gw()** rysującą kolorową gwiazdkę oraz **gwgw()** tworzącą całą ozdobę. Ich parametrem będzie szerokość elementu oraz odpowiednio kolor gwiazdki i typ (numer) ozdoby. Żółw rozpoczyna i kończy rysowanie w środku elementu.

Kolejne wiersze rysowanego układu wymagają analizowania liczby, będącej parametrem funkcji, cyfra po cyfrze. W tym celu badamy kolejno resztę z dzielenie liczby przez 10 oraz iloraz całkowity liczby przez 10. By nie powtarzać tych czynności wielokrotnie przy rysowaniu kolejnych rzędów możemy utworzyć listę złożoną z cyfr liczby, przy czym kolejność cyfr na liście powinna być taka sama, jak w badanej liczbie.

Szerokość tworzonego rysunku jest stała i wynosi 560. Wysokość zależy od liczby cyfr w kodzie opisującym układ gwiazdek. Przy wyliczaniu wysokości rysunku należy wziąć pod uwagę odległość między czubkami przeciwległych ramion gwiazdki, czyli skorzystać z wzoru na wysokość w trójkącie równobocznym. Obie te wartości, szerokość i wysokość rysunku, wykorzystujemy do jego wyśrodkowania.

Rozwiązanie w języku Python

```
from turtle import *
from math import sqrt

def gw(a, k):
    pu(); lt(60); bk(a/3); rt(60); pd()
    fillcolor(k);
    begin_fill()
    for i in range(6):
        rt(60); fd(a/3); lt(120); fd(a/3);
    end_fill()
    pu(); lt(60); fd(a/3); rt(60); pd()

def gwgw(a, n):
    # dla danej cyfry liczę resztę z dzielenia przez 4
    # by wybrać jeden z 4 typów ozdób
    n = n % 4
    if n>1:
        k = "tomato"
    else:
        k = "olivedrab"
    gw(a, k);
    if (n==1 or n==3):
        gw(a/2, "white")

def gwiazdy(kod):
    szer_rys = 560
    # przygotowuję listę złożoną z cyfr kodu
    cyfry = []
    while kod >0:
        cyfry.append(kod % 10);
        kod = kod // 10
    # odwracam kolejność cyfr na liście
    cyfry = cyfry[::-1]
    # wyliczam szerokość ozdoby
    szer_gw = szer_rys / (len(cyfry))
    # przemieszczam żółwia, by rysunek był wyśrodkowany
    pu(); bk(szer_rys/2 - szer_gw/2); lt(90)
    fd(2* szer_gw*sqrt(3)/3); rt(90); pd()
    #rysuję trzy rzędy ozdób
    for a in range(3):
        # oglądam kolejne cyfry kodu i rysuję odpowiednie gwiazdki
        for i in cyfry:
            gwgw(szer_gw, i)
            pu(); fd(szer_gw); pd()
        # usuwam ostatnią cyfrę liczby - kolejny rząd ma być krótszy
        cyfry.pop()
        # przemieszczam żółwia na początek kolejnego rzędu
        pu(); bk(len(cyfry)* szer_gw); rt(90)
        fd(2* szer_gw*sqrt(3)/3); lt(90); pd()
```

Testy

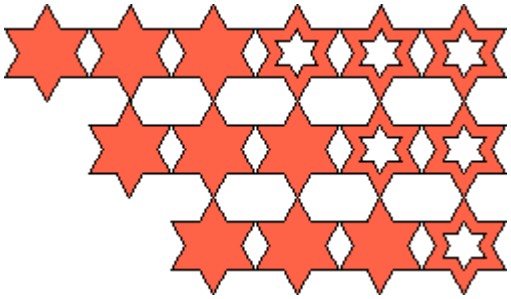
Testowanie rozwiązania rozpoczynamy od przykładów zawartych w treści zadania. Potem testujemy działanie programu dla różnych kodów liczbowych opisujących rysunek – różna liczba cyfr, wszystkie możliwe cyfry (ozdoby), różna kolejność cyfr, cyfry powtarzające się itp. Sprawdzamy, czy szerokość rysunku jest prawidłowa oraz czy jest on wyśrodkowany.

W języku Python, aby przyspieszyć tworzenie rysunku przez żółwia, stosujemy wywołanie złożone z funkcji **tracer()** – rysownie w pamięci, właściwego wywołania funkcji **gwiazdy()** i na końcu uaktualniamy ekran za pomocą funkcji **update()**. Przykład:

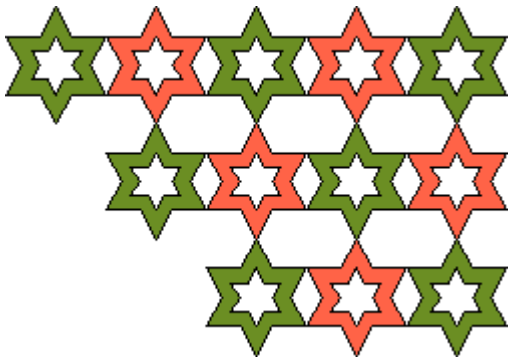
```
tracer(0)
gwiazdy(1234567890)
update()
```

Powrót do standardowego trybu rysowania uzyskamy wywołując funkcję **tracer()** z parametrem równym 1.

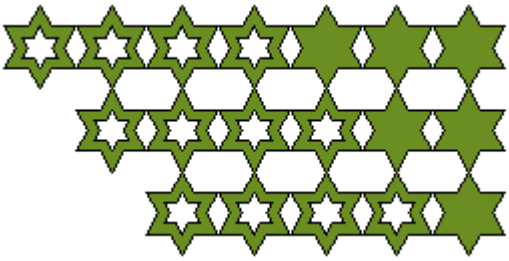
Przykładowe testy:



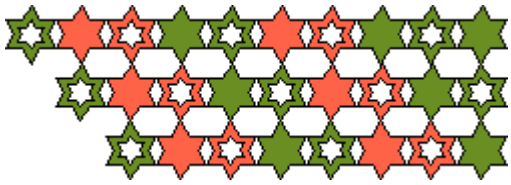
gwiazdy 262333



gwiazdy 97531



gwiazdy 1591408



gwiazdy 1234567890

Zadanie Liczby lustrzane – LOGIA 19 SP (2018/19), etap 2

Treść zadania

Liczby lustrzane to takie dwie liczby, które są swoim lustrzanym odbiciem, np.: 125 i 521. Adam zauważył, że bardzo często ich suma jest liczbą palindromiczną, czyli taką, którą czyta się tak samo od lewej i prawej. Na przykład 125 + 521 = 646. Jeżeli otrzymana suma nie jest liczbą palindromiczną, wtedy Adam dodaje do niej jej liczbę lustrzaną i ponownie sprawdza, czy nie jest liczbą palindromiczną. Wykonuje maksymalnie **n** prób, chyba że wcześniej otrzyma liczbę palindromiczną.

Zdefiniuj dwuparametrową funkcję **lustro**, której pierwszym parametrem jest dodatnia liczba całkowita nie większa niż 100 000, a drugim maksymalna liczba prób wykonywanych przez Adama (od 1 do 8). Wynikiem jest liczba palindromiczna powstała według reguły Adama lub -1, gdy nie da się jej uzyskać w podanej liczbie prób.

Przykłady:

wynikiem **lustro(125,3)** jest **646**

wynikiem **lustro(91,2)** jest **121**

wynikiem **lustro(91,1)** jest **-1**

Omówienie rozwiązania

Rozwiązanie zadania polega na powtórzeniu, co najwyżej **n** razy, obliczenia sumy dwóch liczb, pierwszą z nich jest liczba podana jako parametr, drugą jej lustrzane odbicie. Jeśli otrzymana suma jest liczbą palindromiczną, czyli taką, którą czyta się tak samo od lewej i prawej, to jej wartość jest wynikiem działania funkcji. W przeciwnym wypadku, czyli wtedy, gdy otrzymana suma nie jest liczbą palindromiczną należy sprawdzić, czy liczba wykonanych operacji dodawania jest równa drugiemu parametrowi **n**:

- jeśli tak, to wynikiem działania funkcji jest -1,
- jeśli nie, to powtarzamy wszystko od początku, zamiast parametru do obliczeń bierzemy ostatnio otrzymaną sumę.

OEiiZK
Ośrodek Edukacji Informatycznej
i Zastosowań Komputerów w Warszawie

Rozwiązanie w języku Python

Operacja dodawania musi być wykonana przynajmniej jeden raz, co wynika z treści zadania. Liczba powtórzeń tej operacji nie jest znana, dlatego w takim przypadku warto zastosować pętlę **while**, w której obliczamy sumę tak długo, dopóki nie otrzymamy w wyniku liczby palindromicznej lub liczba powtórzeń wyniesie **n**. Pamiętamy, że **return** kończy działanie funkcji niezależnie od liczby powtórzeń.

```
1.  def czy_palindrom(liczba):
2.      return str(liczba) == str(liczba)[::-1]
3.
4.  def l2(liczba):
5.      return int(str(liczba)[::-1])
6.
7.  def lustro(liczba,n):
8.      i=0
9.      nowa = liczba + l2(liczba)
10.     while i<n:
11.         if czy_palindrom(nowa):
12.             return nowa
13.         else:
14.             nowa = nowa + l2(nowa)
15.             i+=1
16.     return -1
```

Testy

Podczas testowania warto sprawdzić następujące przypadki:

- dla danego parametru wynikiem jest liczba palindromiczna w pierwszym dodawaniu
- brak wyniku dla określonej drugim parametrem liczby prób
- wynik w postaci liczby palindromicznej dla dokładnie n prób
- wynik w postaci liczby palindromicznej dla liczby prób mniejszej niż wartość drugiego parametru

Przykładowe wywołanie testu:

```
print(lustro(323, 1))
```

Wywołanie – Python	**Wynik**
lustro(323, 1)	646
lustro(42, 1)	66
lustro(298, 1)	-1
lustro(987, 3)	-1
lustro(98319, 5)	-1
lustro(98319, 6)	189444981
lustro(98319, 7)	189444981
lustro(4, 5)	8
lustro(5, 2)	11
lustro(66656, 2)	355553
lustro(9899, 8)	22399322

Zadanie Szyfr Vigenère'a – LOGIA 19 SP (2018/19), etap 2

Treść zadania

Ania do szyfrowania wiadomości wykorzystuje tabelę liter (rysunek obok) oraz klucz. Każdej literze tekstu jawnego przyporządkowuje literę z tabeli znajdującą się na przecięciu wiersza wyznaczanego przez tę literę i kolumny odpowiadającej kolejnej literze klucza. Jeżeli długość klucza jest mniejsza niż długość tekstu szyfrowanego, to powiela klucz.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Przykład dla klucza **LOGIA** i tekstu szyfrowanego **OLAMAKOTA**:

tekst jawny: `OLAMAKOTA`

klucz: `LOGIALOGI`

szyfrogram: `ZZGUAVCZI`

Zdefiniuj dwuparametrową funkcję **deszyfr**, której parametrami są dwa słowa o długości od **1** do **1000** złożone z wielkich liter alfabetu łacińskiego, odpowiednio szyfrogram i klucz. Wynikiem jest słowo będące odszyfrowanym tekstem jawnym.

Przykłady:

Wynikiem **deszyfr("ZZGUAVCZI", "LOGIA")** jest **"OLAMAKOTA"**.

Wynikiem **deszyfr("CGSMURRBO", "KRET")** jest **" SPOTKANIE "**.

Omówienie rozwiązania

Podczas analizy działania szyfru Vigenère'a opartego na tablicy z treści zadania należy zauważyć, że każdy z wierszy odpowiada szyfrowi Cezara, przy czym w pierwszym wierszu przesunięcie wynosi 0, w drugim 1 itd. Szyfr Cezara jest to rodzaj szyfru podstawieniowego, w którym każda litera tekstu jawnego (niezaszyfrowanego) zastępowana jest inną, oddaloną od niej o stałą liczbę pozycji w alfabecie. Zakładamy, że alfabet „zawija się" i za literką Z następuje znów litera A.

Do szyfrowania można użyć kodu ASCII, który przyporządkowuje każdej literze liczbę, na przykład kod ASCII litery **A** to **65**, litery **K** to **75**, a litery **Z** to **90**.

OEiiZK
Ośrodek Edukacji Informatycznej
i Zastosowań Komputerów w Warszawie

Odszyfrowanie tekstu to tak naprawdę ponowne zaszyfrowanie z kluczem będącym jego dopełnieniem do 26 (liczba liter alfabetu łacińskiego.). Jeśli tekst jawny zaszyfrowano z kluczem 12, to odszyfrowanie polega na zaszyfrowaniu kryptogramu kluczem 26 – 12 = 14, co oznacza, że trzeba przesunąć się do przodu o 14 liter.

Rozwiązanie w języku Python

Aby otrzymać kod ASCII danego znaku, należy wykorzystać funkcję **ord(znak)** – na przykład wynikiem **ord("K")** będzie **75**. Odwrotna funkcja to **chr(kod)**. Pozwala ona otrzymać znak odpowiadający danemu kodowi ASCII – na przykład wynikiem **chr(90)** będzie **Z**.

Aby zaszyfrować jeden znak danym kluczem należy zamienić literę na kod ASCII, odjąć od niej kod ASCII litery A, a następnie dodać klucz. Kolejne działanie to obliczenie reszty z dzielenia wyniku przez 26, dodanie kodu ASCII litery A i zamiana całości na literę.

Funkcja **ktora(litera)** daje w wyniku numer litery od 0 do 25, który jest kluczem szyfru Cezara. Dlatego do odszyfrowywania używamy klucza będącego różnica 26 i numeru wiersza.

```
1.  def szyfruj_znak(znak, klucz):
2.      return chr((ord(znak) - ord('A') + klucz) % 26 + ord('A'))
3.
4.  def ktora(litera):
5.      return ord(litera)-ord('A')
6.
7.  def deszyfr(tekst, klucz):
8.      pom = ""
9.      dl = len(klucz)
10.     for i in range(len(tekst)):
11.         k = ktora(klucz[i % dl])
12.         pom = pom + szyfruj_znak(tekst[i], 26 - k)
13.     return pom
```

Testy

```
deszyfr("L","K")
```

wynik "B"

```
deszyfr("H","S")
```

wynik "P"

```
deszyfr("UABMOYDRTSQUDFPPOFLBVWWWYHTGLBAEC","HASLO")
```

wynik "NAJBARDZIEJULUBIONANOWELKATOANTEK"

```
deszyfr("GASOEGUYMDYIFZOMHNJASLVKOZICUSGID","TAJNEPRZEZPOUFNE")
```

wynik "NAJBARDZIEJULUBIONANOWELKATOANTEK"

```
deszyfr("KPVHOPUAMTUZFRYZEKWDEJPBIJ","KOTEK")
```

wynik "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

```
deszyfr("KPVHOPUAMTUZFRYZEKWDEJPBIJ","KOTEKKOTEKKOTEKKOTEKKOTEKKKKKKKK")
```

wynik "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

```
deszyfr("A","KLUCZTOZABEZPIECZENIETEGOSZYFRU")
```

wynik "Q"

```
deszyfr("CKSHQHKZNJIITAXEHIXIAXMFOOHCWRNKUYOMBQD","KLUCZTOZABEZPIECZENIETEGOSZY
FRU")
```

wynik "SZYFROWANIEJESTCIEKAWEIZAWIERATAJEMNICE"

```
deszyfr("ACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSR
TVUWYXZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRT
VUWYXZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTV
UWYXZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVU
WYXZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUW
YXZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWY
XZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYX
ZBACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYXZ
BACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYXZB
ACEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYXZBA
CEDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYXZBAC
EDFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKMLNPOQSRTVUWYXZBACE
DFHGIKJLNMOQPRTSUWVXZYACBDFEGIHJLKMONPRQSUTVXWYAZBDCEGFHJIKML","ABC")
```

wynik
"ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZA
BCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZAB
CDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABC
DEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCD
EFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDE
FGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEF
GHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFG
HIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGH
IJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHI
JKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJ
KLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJK
LMNOPQRSTUVWXYZABCDEFGHIJKLMNOPQRSTUVWXYZABCDEFGHIJKL"