Zadatci za vježbu

RJEŠENJA

Zadatak 1.

```
File Edit Format Run Options Windows Help
# 07_01
def cezar(s):
    s2 = ''
    for t in s:
        s2 += chr((ord(t) - 65 - 3) % 26 + 65)
    return s2

s = input()
print(cezar(s))
```

Zadatak 2.

```
File Edit Format Run Options Windows Help
# 07_02
def pomak(s, k):
    s2 = ''
    for t in s:
        s2 += chr((ord(t) - 65 - k) % 26 + 65)
    return s2

s = input()
n = int(input())
print(pomak(s, n))
```

Zadatak 3.

```
File Edit Format Run Options Windows Help
# 07_03
def pomak(s, k):
    s2 = ''
    for t in s:
        s2 += chr((ord(t) - 65 - k) % 26 + 65)
    return s2

s = input()
for i in range(26):
    print(pomak(s, i))
```

Zadatak 4.

```
File Edit Format Run Options Windows Help
# 07_04
##Uključujemo modul za rad s grupom Zn
from O7Z import *

def dek(s, a, b):
    a = Z(26, a)
    b = Z(26, b)
    k = a ** -1
    l = -b / a
    s2 = ''
    for c in s:
        t = k * Z(26, ord(c) - 65) + l
        s2 += chr(int(t) + 65)
    return s2

s = input()
a = int(input())
b = int(input())
print(dek(s, a, b))
```

Zadatak 5.

```
File Edit Format Run Options Windows Help
# 07_05
def dek(s, k):
    s2 = ''
    k *= len(s) // len(k) + 1
    for i in range(len(s)):
        n = (ord(s[i]) - ord(k[i])) % 26
        s2 += chr(n + 65)
    return s2

s = input()
k = input()
print(dek(s, k))
```

Zadatak 6.

```
File Edit Format Run Options Windows Help
# 07_06
def dek(s, k):
    s2 = ''
    for i in range(len(s)):
        n = (ord(k[i]) - ord(s[i])) % 26
        s2 += chr(n + 65)
    return s2

s = input()
k = input()
kljuc = dek(s, k)
##U dobivenom se ključu osnovni ključ možda ponavlja više puta.
##U nastavku tražimo osnovni ključ
i = 0
while i < len(kljuc) and kljuc[:i] in kljuc[i:]:
    i += 1
if i > 1:
    i -= 1
    t = kljuc[i:].index(kljuc[:i])
    print(kljuc[:t + i])
else:
    print(kljuc)
```

Zadatak 7.

```
File Edit Format Run Options Windows Help
# 07_07
def dek(s, k):
    s2 = ''
    k *= len(s) // len(k) + 1
    for i in range(len(s)):
        n = (ord(s[i]) - ord(k[i])) % 26
        s2 += chr(n + 65)
    return s2

s = input()
t = input()

##Isprobavamo sve moguće kombinacije od 3 slova kao ključeve te
##otključavamo kriptirani tekst (funkcija dek) i provjeravamo
##nalazi li se u dekriptiranom tekstu tražena riječ
for i in range(26):
    for j in range(26):
        for k in range(26):
            kljuc = chr(i + 65) + chr(j + 65) + chr(k + 65)
            jasni = dek(s, kljuc)
            if t in jasni:
                print(jasni)
                print(kljuc)
```

Zadatak 8.

```
File Edit Format Run Options Windows Help
# 07_08
from matrica import *
def stupcasta(t, p):
    r = len(t) // len(p)
    s = len(p)
    a = Matrica(r, s)
    for j in range(s):
        for i in range(r):
            a[i, j] = t[j * r + i]
    b = Matrica(r, s)
    for i in range(r):
        for j in range(s):
            b[i, j] = a[i, p[j] - 1]
    s2 = ''
    for i in range(r):
        for j in range(s):
            s2 += b[i, j]
    return s2

s = input()
n = int(input())
p = []
for i in range(n):
    p.append(int(input()))
print(stupcasta(s, p))
```

Zadatak 9.

```
File  Edit  Format  Run  Options  Windows  Help
# 07_09
from matrica import *

class Matrica2(Matrica):
    def __init__(self, r = 1, s = 1):
        super().__init__(r, s)
        return

    def indeksStupca(self, st):
        for i in range(self.m):
            ok = True
            for j in range(self.n):
                if self[j, i] != st[j]:
                    ok = False
            if ok:
                return i
        return -1

    def vratiStupac(self, i):
        a = []
        for j in range(self.n):
            a.append(self[j, i])
        return a

##Funkcija za zadani jasni i otvoreni tekst te za veličinu matrice
##provjerava postoji li permutacija stupaca matrice otvorenog teksta
##koja odgovara matrici kriptiranog teksta
def provjeriDuljinuKljuca(jasni, kriptirani, n):
    s = n
    r = len(kriptirani) // n
    a = Matrica2(r, s)
    b = Matrica2(r, s)
    for i in range(r):
        for j in range(s):
            a[i, j] = jasni[i * s + j]
    for j in range(s):
        for i in range(r):
            b[i, j] = kriptirani[j * r + i]
    OK = True
    i = 0
    p = []
    while i < a.m and OK:
        t = a.indeksStupca(b.vratiStupac(i))
        if t < 0:
            OK = False
        else:
            p.append(t + 1)
        i += 1
    if OK:
        return p
    else:
        return None
```

```
jasni = input()
kriptirani = input()
n = len(kriptirani)
##Promatrat ćemo samo one kljuceve čije su duljine djelitelji duljina tekstova
for i in range(1, n // 2):
    if n % i == 0:
        t = provjeriDuljinuKljuca(jasni, kriptirani, i)
        if t != None:
            for k in t:
                print(k)
```

Zadatak 10.

File Edit Format Run Options Windows Help

```
# 07_10
def mod_pot (a, b, n):
    bz = bin(b)[2:]
    p = a
    v = 1
    for i in range(1, len(bz) + 1):
        if (bz[-i] == '1'):
            v = v * p % n
        p = p * p % n
    return v

p = int(input())
q = int(input())
e = int(input())
d = int(input())
m = int(input())
n = p * q
c = mod_pot(m, e, n)
print(c)
```

Zadatak 11.

File Edit Format Run Options Windows Help

```
# 07_11
def mod_pot (a, b, n):
    bz = bin(b)[2:]
    p = a
    v = 1
    for i in range(1, len(bz) + 1):
        if (bz[-i] == '1'):
            v = v * p % n
        p = p * p % n
    return v

p = int(input())
q = int(input())
e = int(input())
d = int(input())
c = int(input())
n = p * q
m = mod_pot(c, d, n)
print(m)
```

Klasa Z

File Edit Format Run Options Windows Help

```
class Z:
    def __init__(self, n, v):
        self._n = n
        self._v = v
        return

    ##Zbrajanje
    def __add__(self, a):
        return Z(self._n, (self._v + a._v) % self._n)

    ##Množenje
    def __mul__(self, a):
        return Z(self._n, self._v * a._v % self._n)

    ##Aditivni inverz
    def __neg__(self):
        return Z(self._n, self._n - self._v)

    ##Oduzimanje
    def __sub__(self, a):
        return Z(self._n, self._v + (-a)._v)

    ##Potenciranje uključujući i multiplikativni inverz
    def __pow__(self, n):
        if (n < 0):
            i = 1
            while i <= self._n and i * self._v % self._n != 1:
                i += 1
            if i <= self._n:
                return Z(self._n, (i ** -n) % self._n)
            else:
                return -1
        else:
            return Z(self._n, self._v ** n % self._n)

    ##Dijeljenje
    def __truediv__(self, a):
        return self * (a ** -1)

    def __repr__(self):
        return str(self._v)

    ##Predefiniranje funkcije int()
    def __int__(self):
        return self._v
```

Klasa Matrica

```
File  Edit  Format  Run  Options  Windows  Help
class Matrica:
    def __init__(self,n,m=0):
        self.n = n
        if m == 0:
            self.m = n
        else:
            self.m = m
        self.mat = dict()
        return

    def __setitem__(self, ključ, vrijednost):
        self.mat[ključ] = vrijednost
        return

    def __getitem__(self, ključ):
        return self.mat.get(ključ, 0)

    def __repr__(self):
        s = ''
        for i in range(self.n):
            for j in range(self.m):
                s = s + '{0:3}'.format(self[i, j])
            s = s + '\n'
        return s

    def __add__(self, druga):
        if self.n != druga.n or self.m != druga.m:
            print('Zbrajati se mogu samo matrice jednakih dimenzija')
            return
        else:
            zbroj = Matrica(self.n, self.m)
            for i in range(self.n):
                for j in range(self.m):
                    zbroj[i, j] = self[i, j] + druga[i, j]
        return zbroj

    def __mul__(self, druga):
        if self.m != druga.n:
            print('''Matrica se može množiti drugom matricom
                  ako je njezin broj redaka jednak broju stupaca druge
                  matrice''')
            return
        else:
            umnožak = Matrica(self.n, druga.m)
            for i in range(self.n):
                for j in range(druga.m):
                    for k in range(self.m):
                        umnožak[i, j] += self[i, k] * druga[k, j]
            return umnožak

    def gornje_trokutasta(self):
        for i in range(self.n):
            for j in range(i):
```

```
                if self[i, j] != 0:
                    return False
        return True

    def transp(self):
        transponirana = Matrica(self.m, self.n)
        for i in range(self.n):
            for j in range(self.m):
                transponirana[i, j] = self[j, i]
        return transponirana
```