Egy elképzelt nyelvben csak az angol ábécé nagybetűi szerepelnek. A nyelv minden szövege úgy épül fel, hogy blokkokból áll: minden blokk egymástól függetlenül

80% valószínűséggel egy olyan 5 hosszú blokk, melyben csak magánhangzók szerepelnek (A, E, I, O, U)

15% valószínűséggel egy olyan 7 hosszú blokk, melyben csak mássalhangzók szerepelnek,

5% valószínűséggel a következő sztring: "KORE".

Valósítsunk meg egy olyan törő algoritmust, mely a fenti nyelvnek egy legalább 200 hosszú szövegének Vigenere-titkosítását vissza tudja fejteni (jó eséllyel). Használhatjuk a 2-3. órán tekintett módszert, ahol a titkos szöveg ismétléseit kellett detekálni. A Vigenere-kulcs egy 3 és 15 közti hosszúságú lista véletlen shiftekkel.

In [3]:
```
import random
import math
import copy
```

In [4]:
```
# karakterek kodolasa
def indexOfLetter(c):
    return ord(c) - 65
def letterWithIndex(i):
    return chr(i+65)
def shiftChar(c, shift):
    i = indexOfLetter(c)
    new = (i + shift) % 26
    return letterWithIndex(new)
```

In [5]:
```
# kulcs eloallitasa
def generateKey():
    L = []
    random_number = random.randint(3, 15)
    for i in range (0,random_number):
        shift_number = random.randint(0, 26)
        L.append(shift_number)
    return L
```

In [6]:
```
# Vigenere titkosítás -> orai kod
def vigenere(s, shiftList):
    ret = ""
    # Current position in the shiftList
    pos = 0
    n = len(shiftList)
    for c in s:
        new = shiftChar(c, shiftList[pos])
        ret += new
        # Add one, restart if list is over
        pos = (pos + 1) % n
    return ret
```

In [7]:
```
# Valoszinuseghez szamgenerator
# Egy random szam generatorral szimulalom a nyelvben valo elofordul
asi valoszinuseget
# 1 es 100 kozott generalok random szamot, es a hatarok reprezental
jak az elofordulasi valoszinuseget 100 esetet tekintve
def blockProbability():
    L = ""
    rand_prob = random.randint(1, 100)
    if ((rand_prob > -1) and (rand_prob < 81)):
        for i in range (0,5):
            rand_char = random.randint(0,4)
            m = ['A', 'E', 'I', 'O', 'U']
            L += m[rand_char]
    elif ((rand_prob > 80) and (rand_prob < 96)):
        for i in range (0,7):
            m = ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M',
'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z']
            rand_char = random.randint(0, len(m)-1)
            L += m[rand_char]
    else:
        L += "KORE"
    return L
```

In [8]:
```
# Titkos szoveg eloallitasa
def encodingText (s, L):
    for i in range (0, len(s)):
        s[i] = vigenere(s[i], L)
        print(s[i])
```

In [9]:
```
# Ismetlodesek kiszurese
def repetitionsInString(s, length = 3):
    n = len(s)
    ret = [(i,j) for i in range(0, n-length) for j in range(i+1, n-
length+1) if s[i:(i+length)] == s[j:(j+length)] ]
    return ret
```

In [10]:
```
s = []
for i in range (0,100):
    s.append(blockProbability())
L = generateKey()
```

In [11]:
```
# egy hosszu stringet keszitek a szoveg blokkokbol
complete = ""
for i in range (0, len(s)):
    complete += s[i]
print(complete)
```

```
AOEOUEAOAIOUOEIUIUEEUIEAUMZDRPRHOEUEOJTXMDKPUIEOUEEAAIIUIUUEEIEOOA
OIAAUAOEUOAAUIEUAEEEEAIEAEUOTJWBVHPKOREOEOOAIUOOESPQXLTCIEOEOEAEOU
FDRTCQVOIUUEOIUIUOEEEIUUEIAAUAUEIUAIIOIEAUNNQLJYLKOREVSJJQDZKOREBV
SQZNMUEAAEAIEAAIUOIEPWHDFTNOIAAOUOIEIAEIAUIEIUOIEAEEIIOAOAEIIIEAEI
AEOUEUAOIAEEOUIEAIIOOOUIUASKOJMXQEAUEUAIOUAIAIAEUUAEOWFQBTOMIEUEOU
UIUIIOAEUEUIOOEUAAUWVNZVORIEOEOAAUUUEEAAEUEAIOKOREUUUEAUOUOUEOIUUL
KMXMMTUIOIOKOFKCFWEAAIUUAEUIHJDZSQHLFKBXBGOUUOIOEUUIOUEOAOAOOEEOAI
EIIIUUOUUAAUOUIOEIEOAOUUIIEIAAIAAIEIEUAEIIAAIAAOIOOAOAIUAIAUEOOIUI
```

In [12]:
```
# titkositas ket fele modon

# 1 - blokkonkent titkositom a szoveget -> minden blokk karaktereih
ez adott shiftet hasznalok
r = copy.deepcopy(s)
encodingText(r, L)
```

```
NYESC
RKOEQ
BEOIQ
HSUIM
HSEEC
ZJDVXKQ
BOUIW
WDXQLDY
HSESC
ROAEQ
VEIYC
ROIIW
BKOMI
NEASM
HYAEC
VOUEM
ROEEQ
RKEYW
GTWFDAY
XYRI
BOOSI
VEOSM
FZQBTML
VOOIW
RKESC
```

SNRXKJE
BSUYM
BSUMC
BOEIQ
HEEMI
NEAYM
VEAMQ
BSEEC
AXQPRRU
XYRI
ICJNYWI
XYRI
OFSUHGV
HOAEM
NSEEI
VEOMM
CGHHNMW
BSAEW
HYIIQ
NOIEC
VOIYW
VOAIM
VSOEW
NOIMQ
RKEMI
RYUIC
NYIEM
RYUMM
NSISW
BEIYI
FUONUQZ
RKUIC
NSOYI
VKIEM
HEAIW
JPQFBHV
VOUIW
HEIYQ
VYAIC
REISW
REAEC
JFNDDHA
VOOIW
NKUYC
ROAEM
HOAMW
XYRI
HEUII
HYUSC
RYIYC
YUMBUFC
HSOMW
XYFOKYF

```
RKAMC
HKEYQ
UTDDAJQ
YPKFFUP
BEUSQ
BOUYQ
BEESI
BKOSM
RYAMM
VSIYC
BEUEI
HYUMW
RSESI
BEUMQ
RSAEQ
NKIIQ
REAIQ
VKAMI
NYISW
NYAMC
NSAYM
BYIYQ
```

In [13]:
```
# 2 - az egesz szovegen egyben vegig megyek, es karakterenkent kodo
lok
complete_text = copy.deepcopy(complete)
vigenere(complete_text, L)
```

Out[13]:
```
'NYESCXJPIVYUSMBDJCROUMMTDNHQBPVPHNVMBTTBUWTQCVOOYMXJBQVEIYCXNJMBY
ASQTJVIBOUSITDJMHKEIMXJJMNOUSBCFCDUZKSZXXFWBKIYWHNTXDHLXKBNPMBOAIW
NOEZGMQZWBDVMBSUMCHNFMVEUIQTJVIHOIYIBRPQRKURVJUKGYUOVMOBKRDNZOWKNC
DFAZRUNNBIRKIIITRVWVOPAPWOUVBSAEWNXJMVKEMINRFQHYIIIXNJQBKOEMBRJMNO
IEMHDFCNYIEMXXVQRKIMWHXVQHKSOWCVYYRKUICTRPCNSAMIXDVIRYWJYUCPUVOUIW
NDJCVSOEMNNVQBYEYITDXDAJVSZBNPMBKAYCNNFINOUIIBXLWEOUYCXJVWHYUIWBDV
TXWXQUMDJWVYKSNDLGERKAMCNJFCVRJHHLZITSUBBJZXVCBSOICNRPCRYASIHXFMBK
IIQBRVCBEUEINXVQBOIIWTXVCVSEMITRBIVOIICTNJQNKIEIHRPWNYAMCTRBCRYOMC
B'
```

In [14]:
```
# ismerem a nyelvet, igy tudom, hogy vannak benne 4 hosszu KORE str
ingek, eloszor ezekkel kezdem
# a 4 hosszu stringeknel pontosan vissza tudom fejteni, hogy az els
o 4 karakter mennyivel van eltolva

four_char = repetitionsInString(complete_text, 7)
print(four_char)
```

```
[]
```

In [15]:
```
def guessedKeyLength(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext,
length)]
    return gcd(differences)
```

In [16]:
```
# megprobalom a 4 hosszu ismetlodesekbol az eltolasokat meghatarozn
i
def KeyLengths(ciphertext, length = 3):
    differences = [j-i for (i,j) in repetitionsInString(ciphertext,
length)]
    return differences
```

In [18]:
```
differences = KeyLengths(complete_text, 5)
print(differences)

def gcdOfKeys(array):
    if len(array) > 0:
        b=array[0]
        for i in range (0, len(array)):
            s=gcd(b,array[i])
            b=s
        return b
print(gcdOfKeys(differences))
```

```
[317, 90, 418, 121, 234, 247, 204, 29, 14, 14]
1
```

In [20]:
```
m = ['A', 'E', 'I', 'O', 'U']
L2 = [3,2,1]
complete2 = copy.deepcopy(complete)
print(complete2)
complete2
complete2 = vigenere(complete2, L)

print(L)
print(complete2)
print(repetitionsInString(complete2, 5))
diff = KeyLengths(complete2, 4)
gcd_needed = gcdOfKeys(diff)

print(gcd_needed)

# adott szovegben valo ismetlesek kiiratasa
print(complete2[67:72])
print(complete2[157:162])
print(complete[67:72])
print(complete[157:162])
print()
print(complete2[122:127])
print(complete2[356:361])
print(complete[122:127])
print(complete[356:361])
```

```
AOEOUEAOAIOUOEIUIUEEUIEAUMZDRPRHOEUEOJTXMDKPUIEOUEEAAIIUIUUEEIEOOA
OIAAUAOEUOAAUIEUAEEEEAIEAEUOTJWBVHPKOREOEOOAIUOOESPQXLTCIEOEOEAEOU
FDRTCQVOIUUEOIUIUOEEEIUUEIAAUAUEIUAIIOIEAUNNQLJYLKOREVSJJQDZKOREBV
SQZNMUEAAEAIEAAIUOIEPWHDFTNOIAAOUOIEIAEIAUIEIUOIEAEEIIOAOAEIIIEAEI
AEOUEUAOIAEEOUIEAIIOOOUIUASKOJMXQEAUEUAIOUAIAIAEUUAEOWFQBTOMIEUEOU
UIUIIOAEUEUIOOEUAAUWVNZVORIEOEOAAUUUEEAAEUEAIOKOREUUUEAUOUOUEOIUUL
KMXMMTUIOIOKOFKCFWEAAIUUAEUIHJDZSQHLFKBXBGOUUOIOEUUIOUEOAOAOOEEOAI
EIIIUUOUUAAUOUIOEIEOAOUUIIEIAAIAAIEIEUAEIIAAIAAOIOOAOAIUAIAUEOOIUI
[13, 10, 0, 4, 8, 19, 9, 1, 8]
NYESCXJPIVYUSMBDJCROUMMTDNHQBPVPHNVMBTTBUWTQCVOOYMXJBQVEIYCXNJMBYA
SQTJVIBOUSITDJMHKEIMXJJMNOUSBCFCDUZKSZXXFWBKIYWHNTXDHLXKBNPMBOAIWN
OEZGMQZWBDVMBSUMCHNFMVEUIQTJVIHOIYIBRPQRKURVJUKGYUOVMOBKRDNZOWKNCD
FAZRUNNBIRKIIITRVWVOPAPWOUVBSAEWNXJMVKEMINRFQHYIIIXNJQBKOEMBRJMNOI
EMHDFCNYIEMXXVQRKIMWHXVQHKSOWCVYYRKUICTRPCNSAMIXDVIRYWJYUCPUVOUIWN
DJCVSOEMNNVQBYEYITDXDAJVSZBNPMBKAYCNNFINOUIIBXLWEOUYCXJVWHYUIWBDVT
XWXQUMDJWVYKSNDLGERKAMCNJFCVRJHHLZITSUBBJZXVCBSOICNRPCRYASIHXFMBKI
IQBRVCBEUEINXVQBOIIWTXVCVSEMITRBIVOIICTNJQNKIEIHRPWNYAMCTRBCRYOMCB
[(67, 157), (122, 356)]
9
QTJVI
QTJVI
IAAUA
IAAUA
()
BNPMB
BNPMB
IEOEO
IEOEO
```

In [33]:

```
# statisztikai elofordulasok
m1 = ['A', 'E', 'I', 'O', 'U']
m2 = ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', '
P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z']
m3 = ["KORE"]
language = [m1, m2, m3]
LP = [0.8, 0.15, 0.05]

X = GeneralDiscreteDistribution(LP)
textL = [language[X.get_random_element()] for _ in range(100)]
text = []
text += textL
#text = text.append(textL)

print(text)
```

```
[['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E',
'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U
'], ['A', 'E', 'I', 'O', 'U'], ['KORE'], ['A', 'E', 'I', 'O', 'U']
, ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E',
'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U
'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E
', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['B', 'C', 'D', 'F',
'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', '
V', 'W', 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O', 'U'], ['KORE'], ['A'
, 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I',
'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], [
'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', '
Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O',
'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A',
'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O
', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A
', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I',
'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['B', 'C', 'D', 'F', 'G', 'H
', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'V', 'W'
, 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O',
'U'], ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O',
'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['KORE'], ['A',
'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O
', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['B
', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q'
, 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O', 'U
'], ['A', 'E', 'I', 'O', 'U'], ['KORE'], ['A', 'E', 'I', 'O', 'U']
, ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['B', 'C',
'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', '
S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O', 'U'], ['A'
, 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I',
'O', 'U'], ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N',
'O', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A', 'E',
'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U
'], ['A', 'E', 'I', 'O', 'U'], ['KORE'], ['A', 'E', 'I', 'O', 'U']
, ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E',
'I', 'O', 'U'], ['B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M',
'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A',
'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O
', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['B
', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q'
, 'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z'], ['A', 'E', 'I', 'O', 'U
'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E
', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O',
'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['KORE
'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E
', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O',
'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A',
'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O
', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U'], ['A
', 'E', 'I', 'O', 'U'], ['A', 'E', 'I', 'O', 'U']]
```

In [ ]: