Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

**МЕХАНІКО-ТЕХНОЛОГІЧНИЙ ФАКУЛЬТЕТ**

Кафедра кібербезпеки та програмного забезпечення

**Звіт**

з виконаної лабораторної роботи № 3

Дисципліна: Комп`ютерна логiка

Виконав :

студент академічної групи КІ-15

Аннаєв А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перевірив :

Викладач

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кропивницкий- 2018

Лабараторна работа № 3

**Завдання №1**

Дешифрувати повідомлення, зашифроване шифром бекона

**Вхідні дані**: рядок, передається в програму як аргумент командного рядка. Може містити пробіли та літери латинського алфавіту в будь-якому регістрі.

**Результат роботи**: рядок - дешифроване повідомлення

**Модуль : Francis\_Bacon\_Cipher**

#! Шифр Френсиса Бекона

import string

''' Генерация таблицы для кодирования и декодирования шифра Бекона по ключу '''

def create\_table(key,type):

if not isinstance(key, str) or len(key) < 32:

return 'Error';

encode\_table = {}

decode\_table = {}

alpha = string.ascii\_lowercase;

for i in range(len(alpha)):

encode = {alpha[i]: str(key[i:i + 5])}

encode\_table.update(encode);

decode = {str(key[i:i + 5]):alpha[i]}

decode\_table.update(decode);

if type == 'encode':

return encode\_table;

elif type == 'decode':

return decode\_table;

''' Раскодирование шифра Бекона '''

def decoder(text,key):

converted\_text = ''

logic\_conclus\_len = 0;

decoded\_list = []

decoded\_text = ''

resource = create\_table(key, 'decode')

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Определение текста который несет только закодированный текст\*\*\*\*

text = text.replace(' ','');

text\_len = len(text);

# Определение остатка логического завершения текста

logic\_conclus\_len = text\_len%5;

# Разделения текста от остатка

text = text[:text\_len - logic\_conclus\_len];

# Конвертирование текста в представление ab группы

for char in text :

if char.isupper():

converted\_text = converted\_text + 'b'

else :

converted\_text = converted\_text + 'a'

# \*\*\*\*\*\*\*\*\*\*\*\*Представление текста в виде списка по 5 символов\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for i in range(0, len(converted\_text), 5): # 2

decoded\_list.append(converted\_text[i:i + 5]);

# \*\*\*\*\*\*\*\*\*\*\*\*\* Расшифровка текста\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for item in decoded\_list:

if item in resource:

decoded\_text = decoded\_text + resource.get(item);

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Результат расшифрованный текст\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

return decoded\_text;

''' Шифрация текста при помощи шифра Бекона '''

def encoder(text, sentence, key):

logic\_conclus\_ofsentecte = ''

converted\_text = '' # Представление текста в "аb" формате

text = text.lower(); # Перевод текста в нижний регистр

need\_len\_toencode = 0; # Количество символов необходимых для сохранения кода

coded\_text = ''

help\_iteator = 0;

# Набор правил шифрации

resource = create\_table(key, 'encode')

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Преобразование текста в код Френсиса \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for char in text:

if char in resource:

converted\_text = converted\_text + resource.get(char);

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Определение длины фразы не теряя пробелов и логики предложения \*\*

while need\_len\_toencode < len(converted\_text):

if sentence[help\_iteator] != ' ':

need\_len\_toencode += 1;

if len(sentence) < help\_iteator :

sentence = sentence\*2

help\_iteator += 1;

temp = help\_iteator;

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Определение логичного завершения фразы\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

while sentence[temp] != ' ' or help\_iteator >= len(sentence):

logic\_conclus\_ofsentecte = logic\_conclus\_ofsentecte + sentence[temp];

temp += 1;

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*предложение которое будет преобразовываться\*\*\*\*\*\*\*\*\*\*\*

sentence = sentence[:help\_iteator];

help\_iteator =0;

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Кодировка теста в предложении\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

for i in range(len(sentence)):

# Преобразовываем только буквы

char = str(sentence[i]);

if char.isalpha():

if converted\_text[help\_iteator] == 'a':

coded\_text = coded\_text + char.lower();

help\_iteator += 1;

elif converted\_text[help\_iteator] == 'b':

coded\_text = coded\_text + char.upper();

help\_iteator += 1;

# Остальные символы добавляем в последовательность

else:

coded\_text = coded\_text + char;

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

return coded\_text + logic\_conclus\_ofsentecte;

Test : Task\_1.py

import sys

import Francis\_Bacon\_Cipher as cipher

# Количество аргументов командной строки

arg\_count = len(sys.argv) - 1;

chiper\_key = "aaaaabbbbbabbbaabbababbaaababaab";

if arg\_count == 3 and sys.argv[1] == 'encode':

# text = "Prometheus"

# sentence = "Welcome to the Hotel California Such a lovely place Such a lovely place"

text = str(sys.argv[2]);

sentence = str(sys.argv[3]);

print("Результат: " + cipher.encoder(text, sentence, chiper\_key));

elif arg\_count == 2 and sys.argv[1] == 'decode':

# text = "Hot sUn BEATIng dOWN bURNINg mY FEet JuSt WalKIng arOUnD HOt suN mAkiNG me SWeat"

text = str(sys.argv[2]);

print("Результат: " + cipher.decoder(text, chiper\_key));

**Завдання №2**

Iндіана джонс та перстень цзунь-си

Модуль контроллер:

''' Управление роботом '''

def maze\_controller(runner):

while not (runner.found()):

if runner.go():

if runner.found() == False:

runner.turn\_right()

if runner.go():

if runner.found() == False:

runner.go()

runner.go()

else:

runner.turn\_left()

runner.turn\_left()

if runner.go():

runner.found()

else:

runner.turn\_right()

else:

runner.turn\_left()

if runner.go():

runner.found()

else:

runner.turn\_right()

runner.turn\_right()

Test : Task\_2.py

# определение лабиринта

maze = maze\_example5;

maze\_runner = runner.MazeRunner(maze['m'], maze['s'], maze['f'])

controller.maze\_controller(maze\_runner)

print (maze\_runner.found());