МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра кібербезпеки

**Звіт до лабораторної роботи № 6-7**

на тему “ Програмна реалізація методу гамування. Частина 1-2”

Виконав студент(ка) *Борщ Дмитро*

Варіант *№ 1*

Група КБ-01

Перевірила Лаврик Т.В

**Суми 2022**

**ЗВІТ 6-7**

**Завдання 1.** (**15 б.)** Написати програму, яка буде:

1. Генерувати псевдовипадкову послідовність чисел для гами за допомогою лінійного конгруентного генератора ПСП на основі введених значень параметрів b, A, C та T0.

Умови для чисел, що вводяться: C – непарне і взаємно просте з M; A mod 4 = 1; T0 – початкова величина, вибрана в якості початкового числа. Довжина послідовності визначається довжиною тексту, що шифруються.

2. Виконувати шифрування та дешифрування методом гамування тексту, введеного користувачем.

3. Виводити на екран такі результати:

* значення b, A, C, M та T0;
* згенеровану послідовність (гаму) у десятковому форматі;
* відкритий текст у символьному й десятковому форматі;
* зашифрований текст у символьному й десятковому форматі.

Програма повинна забезпечувати зручний інтерфейс користувача. Усі дані, що вводяться та виводяться повинні супроводжуватись чіткими та зрозумілими для користувача поясненнями.

Текст програми

**import** **sys**

alphabet = [

' ', 'а', 'б', 'в', 'г', 'ґ', 'д', 'е', 'є', 'ж', 'з', 'и', 'і', 'ї', 'й',

'к', 'л', 'м', 'н', 'о', 'п', 'р', 'с', 'т', 'у', 'ф', 'х', 'ц', 'ч',

'ш', 'щ', 'ь', 'ю', 'я'

]

**def** **toBin**(input, b): # Converting string to binary string

**if** str(input).isdigit():

str\_i = str(bin(int(input)))

**else**:

str\_i = str(bin(alphabet.index(input)))

str\_i = str\_i[**2**:]

**while** len(str\_i) < b: str\_i = '0' + str\_i

**return** str\_i

**def** **convertStrToBin**(input, b): # Converting Array to binary

output = []

**for** i **in** input:

output.append(toBin(i, b))

**return** output

**def** **convertBintoStr**(input): # Converting binary array to string

output = []

**for** i **in** input:

output.append(alphabet[int(i, **2**)])

**return** output

**def** **generate**(A, C, b, T\_0, i): # Generating pseudo random numbers

output = []

output.append(T\_0)

**for** i **in** range(i):

generated = (A \* output[-**1**] + C) % pow(**2**, b)

**while** generated > **33**: generated -= **33**

output.append(generated)

**return** output[**1**:]

**def** **crypt**(input, gamma): # Encrypting/Decrypt using gamma method

output = []

**for** i **in** range(len(input)):

first\_number = input[i]

second\_number = gamma[i]

summ = str()

**for** j **in** zip(first\_number, second\_number):

summ += str((int(j[**0**]) + int(j[**1**])) % **2**)

output.append(summ)

**return** output

**def** **main**(): # Main function

**try**:

**if** sys.argv[**1**]: **pass**

**except**:

**print**("Choose flag and try again!!!**\n\t** -e(encrypt)**\n\t** -d(decrypt)")

exit(**0**)

**match** sys.argv[**1**]:

case "-d": # Case of decrypt parameter

encryptedText = str(input("Enter encrypted text: "))

gamma = input("Enter gamma: ").split()

encryptedBin = convertStrToBin(encryptedText, len(gamma[**0**]))

decryptedBin = crypt(encryptedBin, gamma)

decryptedText = convertBintoStr(decryptedBin)

**print**(

"**\n**Your encrypted text is:", encryptedText, "**\n**"

"Its Binary:", encryptedBin, "**\n\n**"

"Gamma is:", gamma, "**\n\n**"

"Decrypted binary:", decryptedBin, "**\n**"

"Decrypted text:", ''.join(decryptedText), "**\n**"

)

case "-e": # Classic encryption mode

A, C, b, T\_0 = map(int, input("Enter A, C, b and T\_0: ").split())

openText = str(input("Enter open text: "))

gamma = generate(A, C, b, T\_0, len(openText))

openTexBin = convertStrToBin(openText, b)

gammaBin = convertStrToBin(gamma, b)

encryptedBin = crypt(openTexBin, gammaBin)

encryptedText = convertBintoStr(encryptedBin)

**print**(

"**\n**Your open text is:", openText, "**\n**"

"Its Binary:", openTexBin, "**\n\n**"

"Generated gamma is:", gamma, "**\n**"

"Its binary:", gammaBin, "**\n\n**"

"Encrypted binary:", encryptedBin, "**\n**"

"Encrypted text:", ''.join(encryptedText), "**\n**"

)

case "-h":

**print**("Choose flag and try again**\n\t** -e(encrypt)**\n\t** -d(decrypt)")

exit(**0**)

case \_:

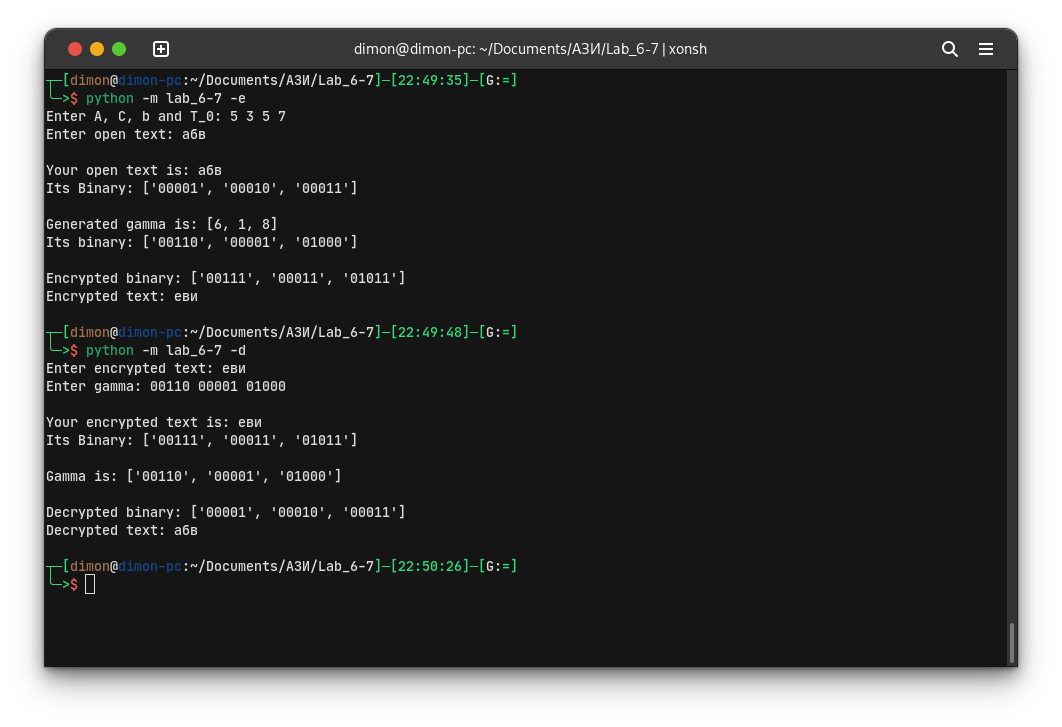
**print**("Wrong flag, try again!!!")

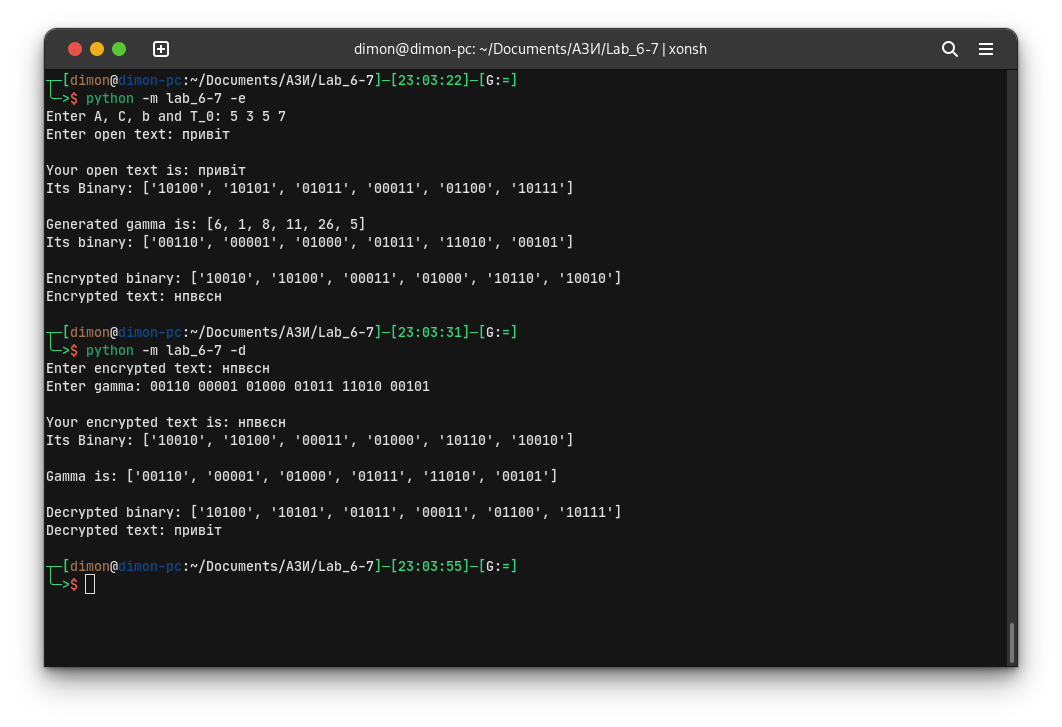
**if** \_\_name\_\_ == "\_\_main\_\_":

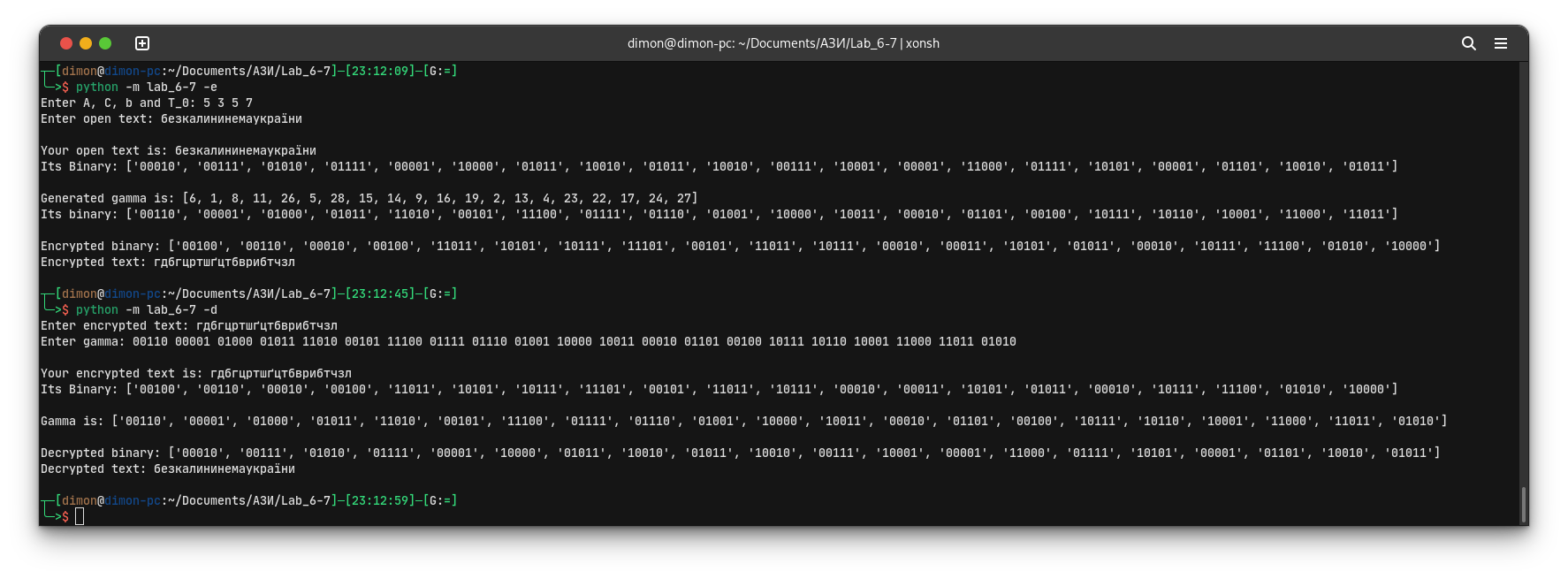
main()

Скріншоти з результатами роботи програми (мінімум 3 різні тексти)

Для першого прикладу візьму дані з прикладу лаборатрної роботи:

Рис. 1 — Перший тест

Рис. 2 — Другий тест зі словом “привіт”.

Рис. 3 — Тест з довгою фразою.