Name: **Mareena Fernandes**
Roll No.: **8669**
Class: **TE IT**
Batch: **B**

EXPERIMENT NO: 1

Vigenere:

```
while True:
    choice = int(input("Enter choice: \n1. Encrypt \n2. Decrypt \n3. Exit\n"))

    if choice == 1:
        plaintext = input("Enter plaintext: ").upper()
        key = input("Enter key string: ").upper()

        if len(key) < len(plaintext):
            times = int(len(plaintext) / len(key))
            while times > 0:
                key += key
                times -= 1

        pt_len = len(plaintext)
        ciphertext = new_plaintext = [0] * pt_len
        pt_len -= 1
        while pt_len > -1:
            ciphertext[pt_len] = chr(
                (ord(plaintext[pt_len]) + ord(key[pt_len])) % 26 + 65
            )
            pt_len -= 1

        ciphertext = "".join(ciphertext)
        print("Ciphertext: ", ciphertext, "\n")

    elif choice == 2:
        ciphertext = input("Enter ciphertext: ").upper()
        key = input("Enter key string: ").upper()

        if len(key) < len(ciphertext):
            times = int(len(ciphertext) / len(key))
```

```
        while times > 0:
            key += key
            times -= 1

    pt_len = len(ciphertext)
    plaintext = [0] * pt_len
    pt_len -= 1
    while pt_len > -1:
        plaintext[pt_len] = chr(
            (ord(ciphertext[pt_len]) - ord(key[pt_len])) % 26 + 65
        )
        pt_len -= 1

    plaintext = "".join(plaintext)
    print("Plaintext: ", plaintext, "\n")
elif choice == 3:
    exit(0)

else:
    print("Invalid choice!\n")
```

Keyed Transposition:

```
while True:
    choice = int(input("Enter choice: \n1. Encrypt \n2. Decrypt \n3. Exit \n"))
    if choice == 1:
        plaintext = input("Enter plaintext: ")
        key = input("Enter key string: ")

        sequence = list(key)
        sequence.sort(key=lambda item: ord(item))

        for letter in key:
            for ele in sequence:
                if ele == letter:
                    sequence[sequence.index(ele)] = key.index(letter)
                    break

        if len(plaintext) % len(key) != 0:
            plaintext += " " * (len(key) - (len(plaintext) % len(key)))

        start = 0
        end = len(key)
        blocks = int(len(plaintext) / len(key))
        ciphertext = [0] * len(plaintext)
        while blocks > 0:
            for i in range(start, end):
                ciphertext[i] = plaintext[start + sequence[i % len(key)]]
            start = end
            end += len(key)
            blocks -= 1
        ciphertext = "".join(ciphertext)
        print("Ciphertext:", ciphertext)

    elif choice == 2:
        ciphertext = input("Enter ciphertext: ")
        key = input("Enter key string: ")

        if len(ciphertext) % len(key) != 0:
            print("Invalid key or ciphertext!")
```

```
        break

    sequence = list(key)
    sequence.sort(key=lambda item: ord(item))

    for letter in key:
        for ele in sequence:
            if ele == letter:
                sequence[sequence.index(ele)] = key.index(letter)
                break

    start = 0
    end = len(key)
    blocks = int(len(ciphertext) / len(key))
    plaintext = [0] * len(ciphertext)
    while blocks > 0:
        for i in range(start, end):
            plaintext[start + sequence[i % len(key)]] = ciphertext[i]
        start = end
        end += len(key)
        blocks -= 1
    plaintext = "".join(plaintext)
    plaintext.strip()
    print("Plaintext:", plaintext)

elif choice == 3:
    exit(0)

else:
    print("Invalid Choice!\n")
```

Product Cipher:

```
# Additive Cipher and Rail Fence Cipher

while True:
    choice = int(input("Enter your choice: \n1. Encrypt \n2. Decrypt \n3. Exit \n"))
    if choice == 3:
        exit(0)
    elif choice != 1 and choice != 2:
        print("Invalid Choice!")
    elif choice == 1:
        plaintext = input("Enter the plaintext: ").upper()
        key = int(input("Enter the numeric key: "))
        ciphertext = [[] for y in range(key)]
        row = 0
        movement = 1

        for i in range(0, len(plaintext)):
            if row == 0:
                movement = 1
            elif row == key - 1:
                movement = -1
            ciphertext[row].append(chr((ord(plaintext[i]) + key) % 26 + 65))
            row += movement

        for i in range(0, len(ciphertext)):
            ciphertext[i] = "".join(ciphertext[i])
        ciphertext = " ".join(ciphertext)

        print("Ciphertext: ", ciphertext)

    elif choice == 2:
        ciphertext = input("Enter the ciphertext: ").upper()
        key = int(input("Enter the numeric key"))
        plaintext = [[] for y in range(key)]
        text_len = len(ciphertext) - key + 1
        row = 0
        movement = 1
        final_text = ""
```

```
ciphertext = ciphertext.split(" ")

for i in range(0, len(ciphertext)):
    for letter in ciphertext[i]:
        if letter == " ":
            plaintext[i].append(" ")
        else:
            plaintext[i].append(chr((ord(letter) - key) % 26 + 65))

for i in range(text_len):
    if row == 0:
        movement = 1
    elif row == key - 1:
        movement = -1
    final_text += plaintext[row][0]
    plaintext[row].pop(0)
    row += movement

print("Plaintext: ", final_text)
```

Post labs:

1. Explain Vernam Cipher:

Ans:
It is a substitution cipher. In this technique, each character of plaintext is assigned a number. For example: with only alphabets, usual assignments are: a → 0, b → 1, c → 2, …… z → 25

To encrypt plaintext, a key of equal length is taken. After key selection each character from plaintext is added to corresponding key character using assigned values. The final value is taken mod with 26 to get ciphertext character.

Example:

Plaintext						Key				
H	E	L	L	O	+	A	P	P	C	E
7	4	11	11	14		0	15	15	11	4

7 % 26 19 % 26 26 % 26 22 % 26 18 % 26

Cipher text:

7	19	0	22	18
H	T	A	W	S

For decryption, we reverse the process that is subtract key characters and then mod with 26.

2. Explain Double Columnar Cipher:

Ans:
It is a transposition cipher. It uses a key matrix for transposition. And alphabetical or numeric key is used to mark columns. The plain text characters are inserted into matrix row by row with last row padded using spaces if necessary. The column heading (key) is used to determine order of columns. The characters are taken from top to bottom in every column in order. This process is then repeated using same or different key. To decipher, the cipher is divided into key lengths and arranged in columns. Then key is used to rearrange them to get the plaintext back.

Example:
Plaintext: I AM MAREENA F Key: CODE

Key	C	O	D	E
R1	I	--	A	M
R2	--	M	A	R
R3	E	E	N	A
R4	--	F	--	--

Cipher text: IAREEN MM AAF