PART A

(PART A: TO BE REFERRED BY STUDENTS)

Experiment No.02

A.1—Aim:
Encrypt the given plain text using the Caesar Cipher encryption algorithm and Implementation of Hill Cipher encryption algorithm.

A.2--- Prerequisite:
1. Fundamentals of encryption and decryption process

A.3--- Outcome:
After successful completion of this experiment students will be able to:

1. Learn and appreciate the operation of classical ciphers
2. Understand the shortcomings of these ciphers

A.4--- Theory:

- The art and science of keeping messages secure is **cryptography.** A message is **plaintext (**sometimes called **cleartext).** The process of disguising a message in such a way as to hide its substance is **encryption.** An encrypted message is **ciphertext.** The process of turning ciphertext back into plaintext is **decryption.**

A **cryptographic algorithm**, also called a **cipher**, is the mathematical function used for encryption and decryption. Both the encryption and decryption operations use a key. The range of possible values of the key is called the **keyspace.**

- $E_K(M) = C$
- $D_K(C) = M$
- $D_K(E_K(M)) = M$

There are two basic types of classical ciphers:

- Transposition ciphers
- Substitution ciphers

A **substitution cipher** is one in which each character in the plaintext is substituted for another character in the ciphertext. The receiver inverts the substitution on the ciphertext to recover the plaintext.

For example: **Caesar Cipher,** in which each plaintext character is replaced by the character three to the right modulo 26 ("A" is replaced by "D," "B" is replaced by "E,"..., "X“ is replaced by "A," "Y" is replaced by "B," and "Z" is replaced by "C") is a simple substitution cipher.

- **Hill Cipher:**

Invented by Lester S. Hill in 1929, it was the first polygraphic cipher in which it was practical (though barely) to operate on more than three symbols at once.

Each letter is represented by a number modulo 26. (Often the simple scheme A = 0, B = 1, ..., Z = 25 is used, but this is not an essential feature of the cipher.) To encrypt a message, each block of *n* letters (considered as an *n*-component vector) is multiplied by an invertible *n* × *n* matrix, again modulus 26. To decrypt the message, each block is multiplied by the inverse of the matrix used for encryption.

The matrix used for encryption is the cipher key, and it should be chosen randomly from the set of invertible *n* × *n* matrices (modulo 26). The cipher can, of course, be adapted to an alphabet with any number of letters; all arithmetic just needs to be done modulo the number of letters instead of modulo 26.

Consider the message 'ACT', and the key below (or GYBNQKURP in letters):

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

Since 'A' is 0, 'C' is 2 and 'T' is 19, the message is the vector:

$$\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$$

Thus the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}$$

Which corresponds to a cipher text of 'POH'. Now, suppose that our message is instead 'CAT', or:

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

This time, the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$$

(PART - B)

(TO BE COMPLETED BY STUDENTS)

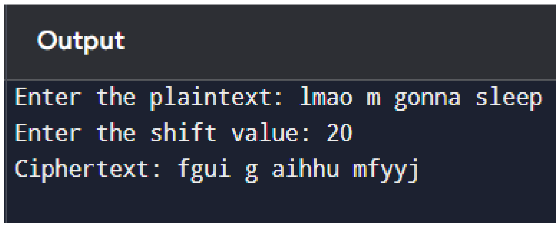
(Students must submit the soft copy as per the following segments within two hours of the practical.The soft copy must be submitted on the portal or on MS Teams before the deadline)

Roll.No. : C175	Name: Nidhish Rathod
Sem/Year : X/5th	Batch: D2
Date of Experiment :	Date of Submission:
Grade --	

B.1: Procedure of performed experiment

- Caesar Cipher

```
#include <iostream>
#include <string>
using namespace std;
string caesarCipherEncrypt(string plaintext, int shift) {
    string ciphertext = "";
    shift = shift % 26;
    for (char c : plaintext) {
        if (isalpha(c)) {
            char offset = isupper(c) ? 'A' : 'a';
            ciphertext += (c - offset + shift) % 26 + offset;
        } else {
            ciphertext += c;
        }
    }
    return ciphertext;
}
int main() {
    string plaintext;
    int shift;

    cout << "Enter the plaintext: ";
    getline(cin, plaintext);
    cout << "Enter the shift value: ";
    cin >> shift;

    string ciphertext = caesarCipherEncrypt(plaintext, shift);
    cout << "Ciphertext: " << ciphertext << endl;

    return 0;
}
```

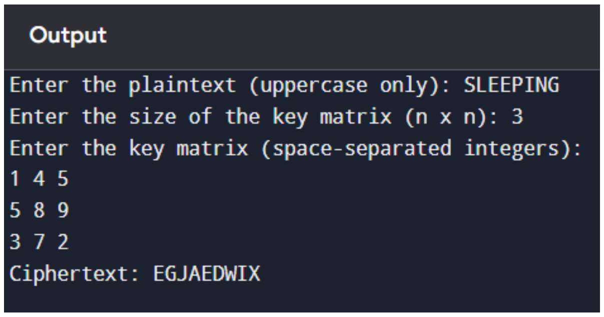

Output

```
Enter the plaintext: lmao m gonna sleep
Enter the shift value: 20
Ciphertext: fgui g aihhu mfyyj
```

- Hill Cipher

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
vector<int> matrixMultiply(vector<vector<int>> key, vector<int> block, int n) {
    vector<int> result(n, 0);
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            result[i] += key[i][j] * block[j];
        }
        result[i] %= 26;
    }
    return result;
}
string hillCipherEncrypt(string plaintext, vector<vector<int>> key, int n) {
    string ciphertext = "";
    vector<int> block(n, 0);

    while (plaintext.size() % n != 0) {
        plaintext += 'X';
    }

    for (size_t i = 0; i < plaintext.size(); i += n) {
        for (int j = 0; j < n; j++) {
            block[j] = plaintext[i + j] - 'A';
        }

        vector<int> encryptedBlock = matrixMultiply(key, block, n);

        for (int val : encryptedBlock) {
            ciphertext += (val + 'A');
        }
    }

    return ciphertext;
}
```

```
int main() {
    string plaintext;
    int n;
    cout << "Enter the plaintext (uppercase only): ";
    cin >> plaintext;
    cout << "Enter the size of the key matrix (n x n): ";
    cin >> n;
    vector<vector<int>> key(n, vector<int>(n));
    cout << "Enter the key matrix (space-separated integers):\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> key[i][j];
        }
    }
    string ciphertext = hillCipherEncrypt(plaintext, key, n);
    cout << "Ciphertext: " << ciphertext << endl;

    return 0;
}
```



```
Output
Enter the plaintext (uppercase only): SLEEPING
Enter the size of the key matrix (n x n): 3
Enter the key matrix (space-separated integers):
1 4 5
5 8 9
3 7 2
Ciphertext: EGJAEDWIX
```

B.2: Observations and Learning's:

Encryption techniques like Caesar Cipher and Hill Cipher demonstrate the principles of classical cryptography, highlighting their simplicity and limitations in ensuring robust security in modern contexts.

B.3: Conclusion:

The implementation of Caesar Cipher and Hill Cipher provides a foundational understanding of classical encryption techniques, emphasizing their operational principles, practical applications, and inherent vulnerabilities, which underline the evolution towards more advanced cryptographic methods.