**CSS**

**EXPERIMENT 1**

Name:Rebecca Dias Class: TECMPNA

Rollno: 19 Pid: 182027

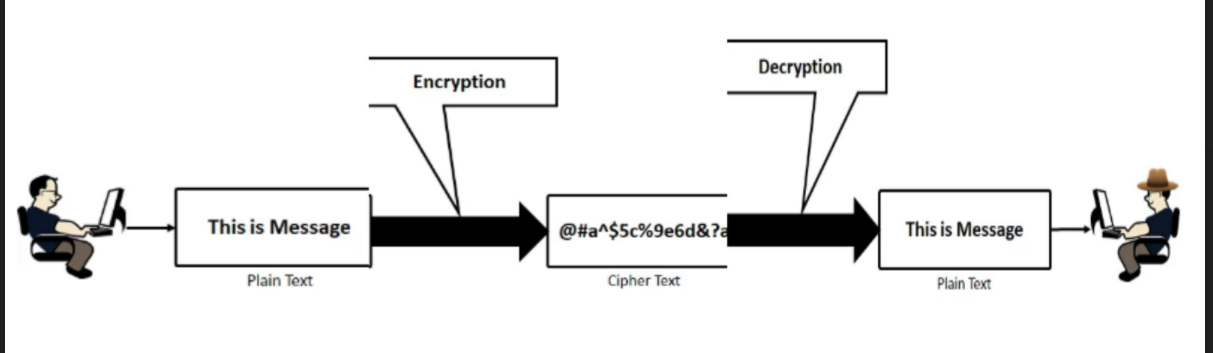
**Aim:** To design and implement Affine cipher and Rail Fence Cipher

**Languages used:** C/Java/Python

**Theory:**

**1. What is cryptography**

* Crypt – ‘hidden’ Graphy – ‘writing’.
* In layman language, hiding information from the outside world and letting only the right receiver know how to see it.
* Cryptography is the technique of securing information and communications through use of codes so that only those people for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information



**2. Types of Classic Cryptosystems**

* Based on types of keys used for encryption and decryption operations, there are two types of cryptography

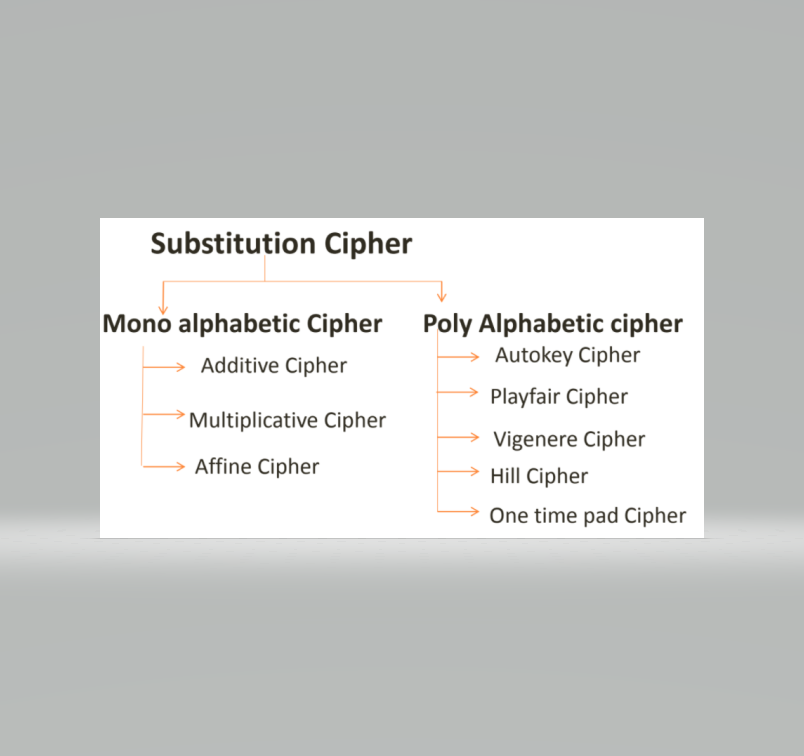
|  |  |
| --- | --- |
| Symmetric Key Cryptography | Asymmetric Key Cryptography |
| Also called Private key Cryptography | Also called as Public-key cryptography |
| Both the sender and receiver will use the same  key for encrypting and decrypting the message. | A public key is used for encryption and a  private key is used for decryption |
|  |  |

**3. Explain Substitution Cipher Technique (Additive, Multiplicative and Affine ) with an example of Affine Cipher [theoretical result and code attached should match].**

* **Substitution Cipher Technique**

1. A substitution cipher replaces one symbol with another.

2. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.



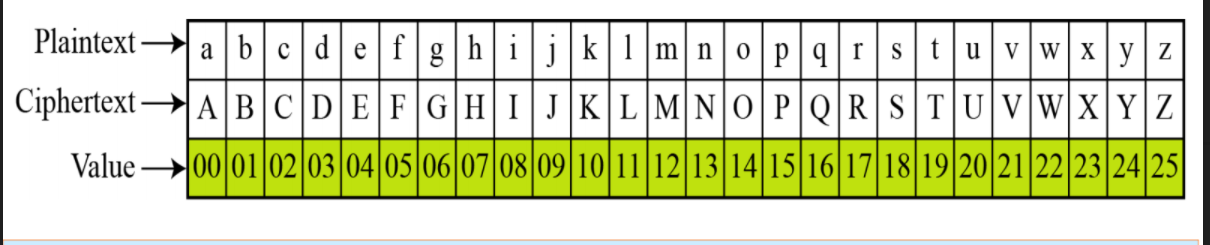
* **Additive Cipher**

1. Also called as SHIFT CIPHER and sometimes as “CAESAR CIPHER”.

2. The term additive cipher better reveals its mathematical nature.

ENCRYPTION → C = (P + k) mod 26 DECRYPTION → P = (C - k) mod 26

Where C=Ciphertext, P=Plaintext, k=key



**When the cipher is additive, the plaintext, ciphertext, and key are integers in Z26.**

* **Multiplicative Cipher**

1. Encryption algorithm specifies multiplication of the plaintext by the key

2. Decryption algorithm specifies division of the ciphertext by the key

3. Key belongs to Z\*26

ENCRYPTION→ C = (P \* k) mod 26

DECRYPTION→ P = (C \* k-1) mod 26

Where C=Ciphertext, P=Plaintext, k=key ,k-1=Multiplicative inverse

**Z\*26 include 1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25**

* **Affine Cipher**

1. It is the combination of ADDITIVE and MULTIPLICATIVE cipher.

2. It uses a combination of both ciphers with a pair of keys.

3. First key --- Multiplicative Cipher

4. Second Key --- Additive Cipher

ENCRYPTION➡ C = (P \* k1 + k2) mod 26

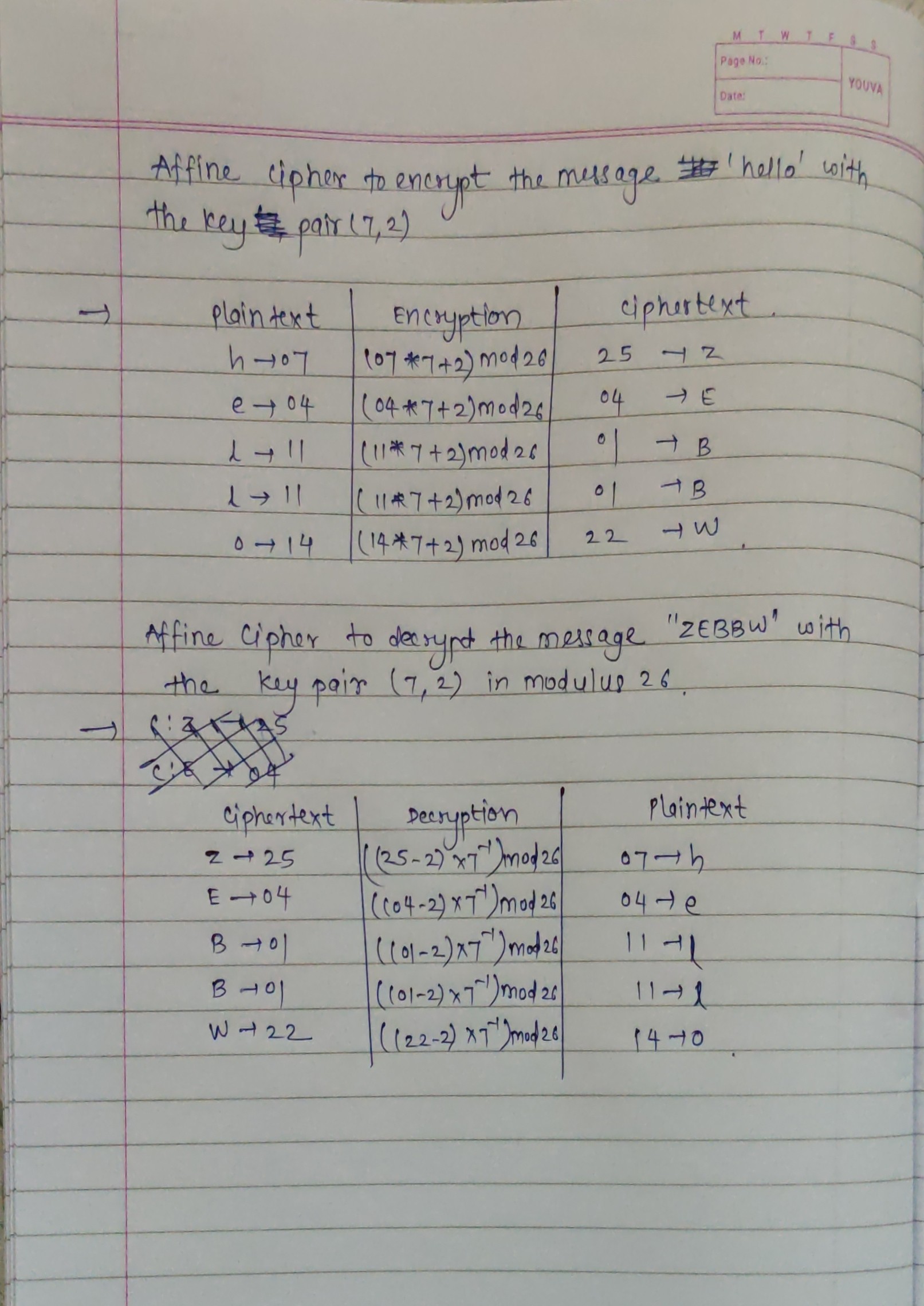
DECRYPTION➡ P = ( [ C- k2 ] \* k-1) mod 26

The Affine Cipher uses a pair of keys in which;

First key is from Z\*26

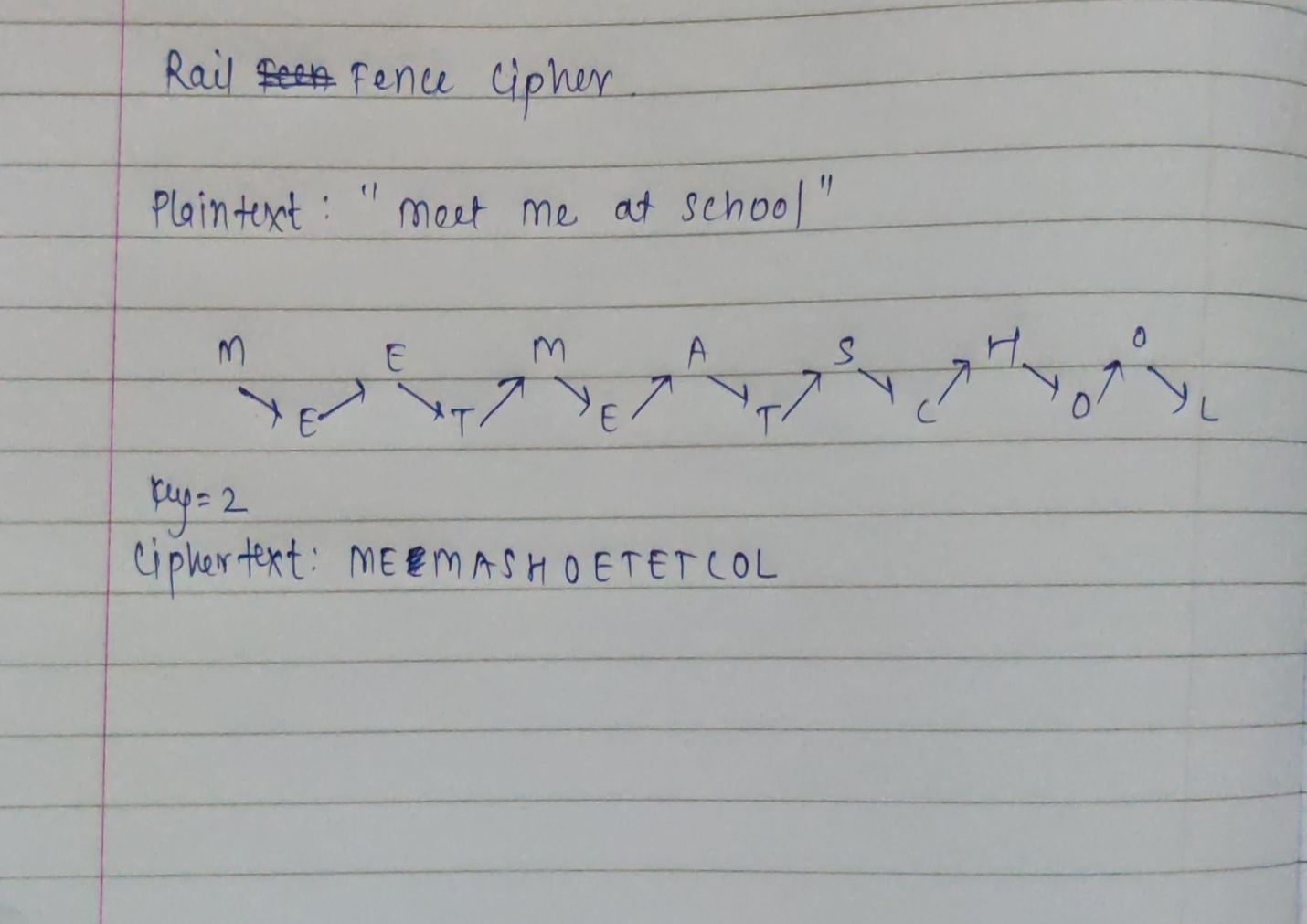
Second Key is from Z26

**Thus, the size of the key domain is 26 \* 12 = 312.**



**4. Explain transposition cipher technique (rail fence) with an example [theoretical result and code attached should match].**

1. In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
2. When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again. Thus the alphabets of the message are written in a zig-zag manner.
3. After each alphabet has been written, the individual rows are combined to obtain the cipher-text



**CODE:**

ch=1

while(ch!=4):

    print("1.Affine Cipher")

    print("2.Railfence Cipher")

    print("3.Exit")

    ch=int(input("Enter your choice :"))

    if(ch==1):

        dict1={"A":0,"B":1,"C":2,"D":3,"E":4,"F":5,"G":6,"H":7,"I":8,"J":9,"K":10,"L":11,"M":12,"N":13,"O":14,"P":15,"Q":16,"R":17,"S":18,"T":19,"U":20,"V":21,"W":22,"X":23,"Y":24,"Z":25}

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

        mulinv={1:3,3:9,5:21,7:15,9:3,11:19,15:7,17:23,19:11,21:5,23:17,25:25}

        lista=[]

        listb=[]

        listd=[]

        liste=[]

        list1=[]

        list2=[]

        list4=[]

        list5=[]

        print("Affine Cipher")

        choice=1

        while(choice!=4):

            print("a)Encryption")

            print("b)Decryption")

            print("c)Exit")

            choice = input("Enter Choice:")

            if(choice == "a"):

                text=input("Enter your plain text:")

                k1=int(input("Enter k1 :"))

                k2=int(input("Enter k2 :"))

                for letter in text:

                     lista.append(letter)

                for a in lista:

                    C=dict1[a]

                    listb.append(C)

                for i in listb:

                    Cipher=(i\*k1+k2)%26

                    listd.append(Cipher)

                for b in listd:

                    X=listc[b]

                    liste.append(X)

                Ciphertext = ' '.join([str(elem) for elem in liste])

                print("Ciphertext:",Ciphertext)

            elif(choice =="b"):

                text=input("Enter yor Cipher  text:")

                k1=int(input("Enter k1 :"))

                k2=int(input("enter k2 :"))

                for letter in text:

                    list1.append(letter)

                for a in list1:

                    C=dict1[a]

                    list2.append(C)

                m=mulinv[k1]

                for i in list2:

                    Encryp=((i-k2)\*m)%26

                    list4.append(Encryp)

                for b in list4:

                    X=listc[b]

                    list5.append(X)

                Ciphertext = ' '.join([str(elems) for elems in list5])

                print("PlainText:",Ciphertext)

            elif(choice == "c"):

                exit()

    elif(ch==2):

        print("Railfence")

        choice=0

        while(choice!=4):

            print("a.Encryption")

            print("b.Decryption")

            print("c.Exit")

            choice = input("Enter Choice:")

            if(choice == "a"):

                def railfence\_enc(plaintext, key):

                    list=[]

                    initialIncrementBy = key + (key - 2)

                    incrementBy = initialIncrementBy

                    for level in range(key):

                        tempList = []

                        i = level

                        while i < len(plaintext):

                            tempList.append(plaintext[i])

                            if len(tempList)%2 == 0 and level!=0 and level!=key-1:

                                if (initialIncrementBy - incrementBy + i < len(plaintext)):

                                    tempList.append(plaintext[ initialIncrementBy - incrementBy + i ])

                                i = initialIncrementBy - incrementBy + i

                            i = i + incrementBy

                        list.extend(tempList)

                        if(level + 1 != key - 1):

                            incrementBy = incrementBy - 2

                        else:

                            incrementBy = key + (key - 2)

                    return list

                plaintext=input("Enter your plaintext: ")

                key=int(input("Enter the key: "))

                print()

                plaintext = plaintext.replace(" ","")

                railfence\_ciphertext=[]

                railfence\_ciphertext=railfence\_enc(plaintext, key)

                ct=""

                ct=ct.join(railfence\_ciphertext)

                print("Ciphertext: "+ct)

            elif(choice =="b"):

                cipher\_text=''.join(input("Enter yor Cipher text :").split())

                key=int(input("Enter the key : "))

                rail = [['\n' for i in range(len(cipher\_text))]

                            for j in range(key)]

                dir\_down = None

                row, col = 0, 0

                for i in range(len(cipher\_text)):

                    if row == 0:

                        dir\_down = True

                    if row == key - 1:

                        dir\_down = False

                    rail[row][col] = '\*'

                    col += 1

                    if dir\_down:

                        row += 1

                    else:

                        row -= 1

                index = 0

                for i in range(key):

                    for j in range(len(cipher\_text)):

                        if ((rail[i][j] == '\*') and

                        (index < len(cipher\_text))):

                            rail[i][j] = cipher\_text[index]

                            index += 1

                result = []

                row, col = 0, 0

                for i in range(len(cipher\_text)):

                    if row == 0:

                        dir\_down = True

                    if row == key-1:

                        dir\_down = False

                    if (rail[row][col] != '\*'):

                        result.append(rail[row][col])

                        col += 1

                    if dir\_down:

                        row += 1

                    else:

                        row -= 1

                plain="".join(result)

                print("Plaintext:",plain)

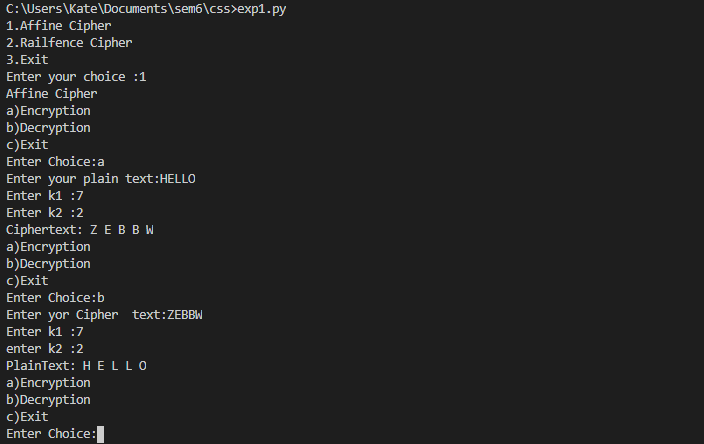
            elif(choice == "c"):

                exit()

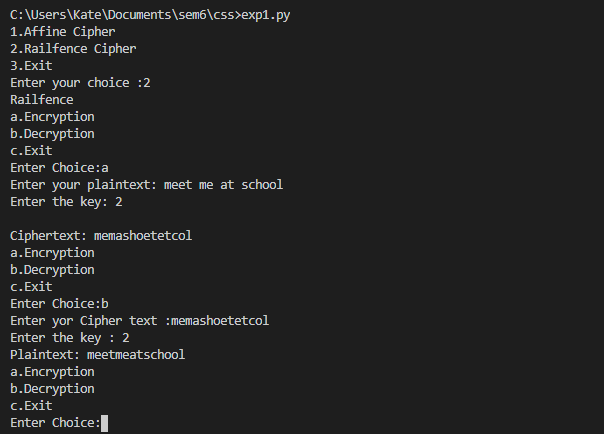
    elif(ch==3):

        exit()

**OUTPUT:**

**AFFINE CIPHER:**

**RAIL FENCE CIPHER:**

****

**Conclusion :** In this experiment we learned about cryptography and the various types of cipher technique.We also implemented affine cipher and rail fence cipher in python and successfully got the output.

**Viva Questions:**

1. **What is public key and private key cryptography**
   1. Public-key cryptography- Both the sender and receiver will use the same key for encrypting and decrypting the message.
   2. Private-key cryptography- A public key is used for encryption and a private key is used for decryption
2. **What is the difference between stream cipher and block cipher**

|  |  |
| --- | --- |
| BLOCK CIPHER | STREAM CIPHER |
| Block Cipher Converts the plain text into cipher text by taking plain text’s block at a time. | Stream Cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time. |
| Block cipher uses either 64 bits or more than 64 bits. | While stream cipher uses 8 bits. |
| The algorithm modes which are used in block cipher are: ECB (Electronic Code Book) and CBC (Cipher Block Chaining) | The algorithm modes which are used in stream cipher are: CFB (Cipher Feedback) and OFB (Output Feedback). |
| Block cipher works on transposition techniques like Caesar cipher, polygram substitution cipher, etc. | While stream cipher works on substitution techniques like rail-fence technique, columnar transposition technique, etc. |
| Block cipher is slow as compared to stream cipher. | While stream cipher is fast in comparison to block cipher. |

1. **Explain playfair cipher with example**

* Was used by British Army during World War I
* The secret key in this cipher is made of 25 alphabet letters arranged in 5\*5 matrix
* Different arrangements of the letters in the matrix can create many different secret keys.

**EXAMPLE:**

**Plain Text:** hello

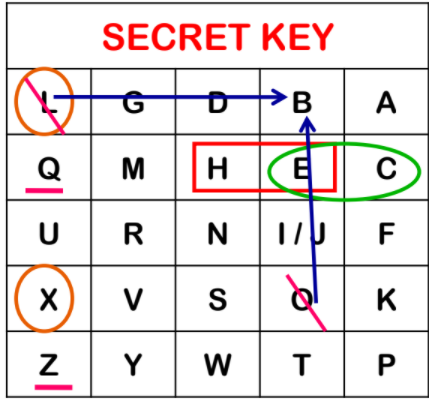
**Inserting Bogus Character**: helxlo

**PAIRING CHARACTERS**:

he lx lo

EC QZ BX

**SECRET KEY:**



**RULES:**

1. If two letters in a pair are located in the same row of the secret key, then the corresponding encrypted character for each letter is the next letter to one right in the same row.
2. If two letters in a pair are located in the same column of the secret key, then the corresponding encrypted character for each letter is the letter beneath it.
3. Not in same row or column, the corresponding encrypted character for each letter is a letter that is in its own row but in the same column as the other letter

**Cipher Text: ECQZBX**

1. **What is the size of key domain of Affine Cipher**

The Affine Cipher uses a pair of keys in which;

First key is from Z\*26

Second Key is from Z26

**Thus, the size of the key domain is 26 \* 12 = 312.**

1. **What is Columnar Transposition**

The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one