Name: Sandeep Sahani Department: TE IT XIE ID: 201903040

EXPERIMENT-1

IMPLEMENTATION OF SUBSTITUTION TECHNIQUE

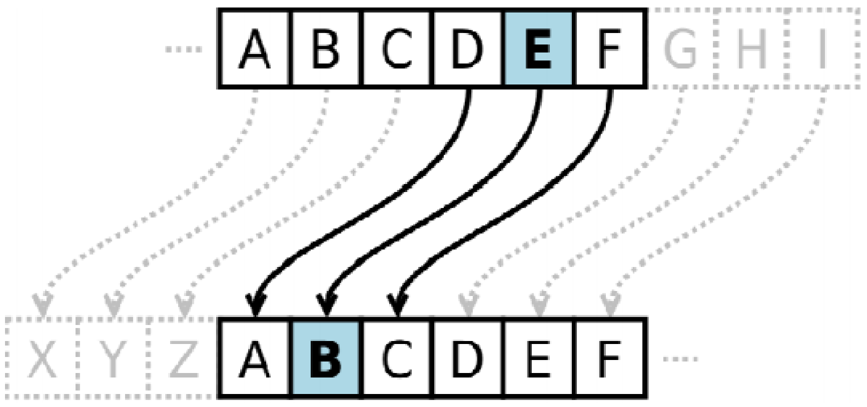
Date of Performance: 23/07/2021
Date of Submission: 13/08/2021

1 a] Ceaser Cipher Encryption and Decryption

Aim:

Theory:
To encrypt a message with a Caesar cipher, each letter in the message is changed using a simple rule: shift by three. Each letter is replaced by the letter three letters ahead in the alphabet. A becomes D, B becomes E, and so on. For the last letters, we can think of alphabet as a circle and "wrap around". W becomes Z, X becomes A, Y becomes B, and Z becomes C. To change a message back, each letter is replaced by the one three before it.

Example:

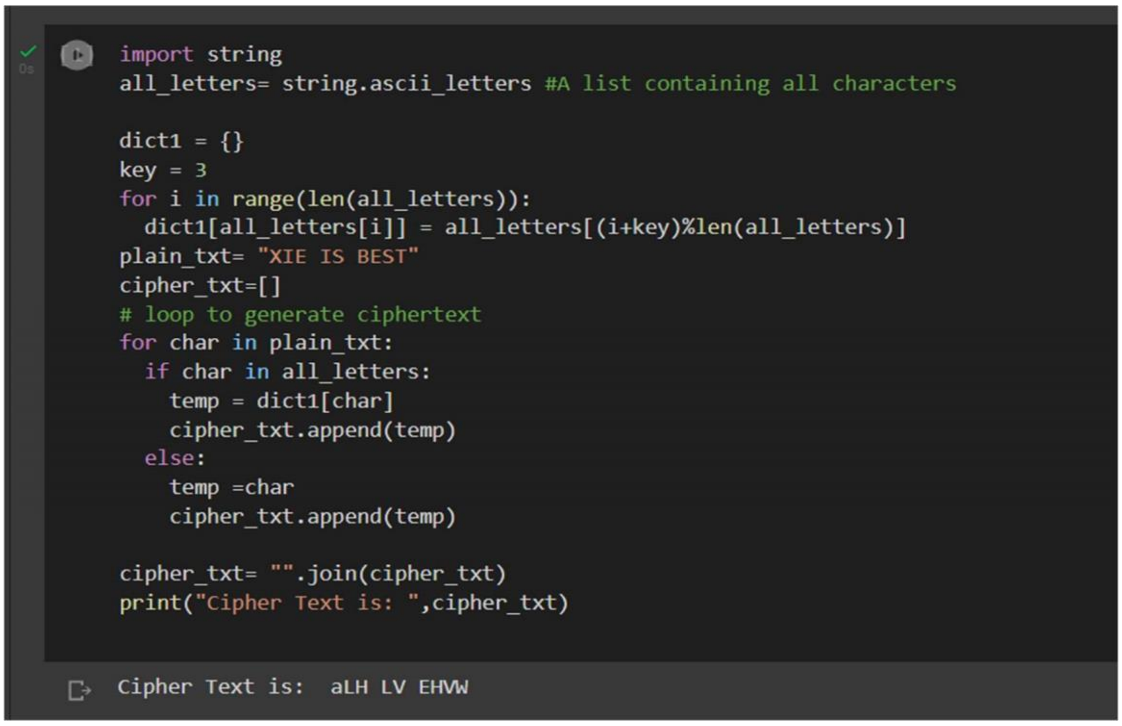


Algorithm:

Code:
```
#ENCRYPTION
import string
all_letters= string.ascii_letters #A list containing all characters

dict1 = {}
key = 3
for i in range(len(all_letters)):
        dict1[all_letters[i]] = all_letters[(i+key)%len(all_letters)]
```

```
plain_txt= "XIE IS BEST"
cipher_txt=[]
# loop to generate ciphertext
for char in plain_txt:
        if char in all_letters:
                temp = dict1[char]
                cipher_txt.append(temp)
        else:
                temp =char
                cipher_txt.append(temp)

cipher_txt= "".join(cipher_txt)
print("Cipher Text is: ",cipher_txt)

#DECRYPTION
dict2 = {}
for i in range(len(all_letters)):
        dict2[all_letters[i]] = all_letters[(i-key)%(len(all_letters))]

# loop to recover plain text
decrypt_txt = []

for char in cipher_txt:
        if char in all_letters:
                temp = dict2[char]
                decrypt_txt.append(temp)
        else:
                temp = char
                decrypt_txt.append(temp)

decrypt_txt = "".join(decrypt_txt)
print("Recovered plain text :", decrypt_txt)
```

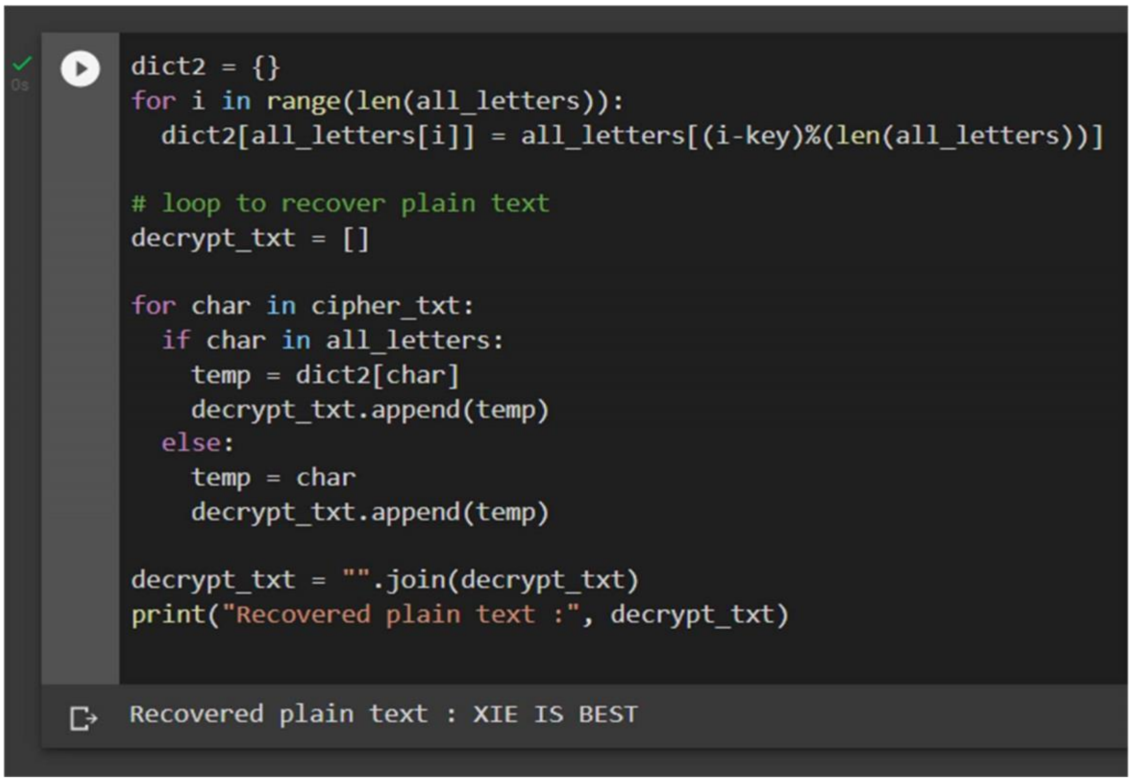
Output:



```
import string
all_letters= string.ascii_letters #A list containing all characters

dict1 = {}
key = 3
for i in range(len(all_letters)):
  dict1[all_letters[i]] = all_letters[(i+key)%len(all_letters)]
plain_txt= "XIE IS BEST"
cipher_txt=[]
# loop to generate ciphertext
for char in plain_txt:
  if char in all_letters:
    temp = dict1[char]
    cipher_txt.append(temp)
  else:
    temp =char
    cipher_txt.append(temp)

cipher_txt= "".join(cipher_txt)
print("Cipher Text is: ",cipher_txt)
```

```
Cipher Text is:  aLH LV EHVW
```

Decrypted Text



```
dict2 = {}
for i in range(len(all_letters)):
  dict2[all_letters[i]] = all_letters[(i-key)%(len(all_letters))]

# loop to recover plain text
decrypt_txt = []

for char in cipher_txt:
  if char in all_letters:
    temp = dict2[char]
    decrypt_txt.append(temp)
  else:
    temp = char
    decrypt_txt.append(temp)

decrypt_txt = "".join(decrypt_txt)
print("Recovered plain text :", decrypt_txt)
```

```
Recovered plain text : XIE IS BEST
```

1 b] Brute force attack on Ceaser Cipher:

Aim:

Theory:

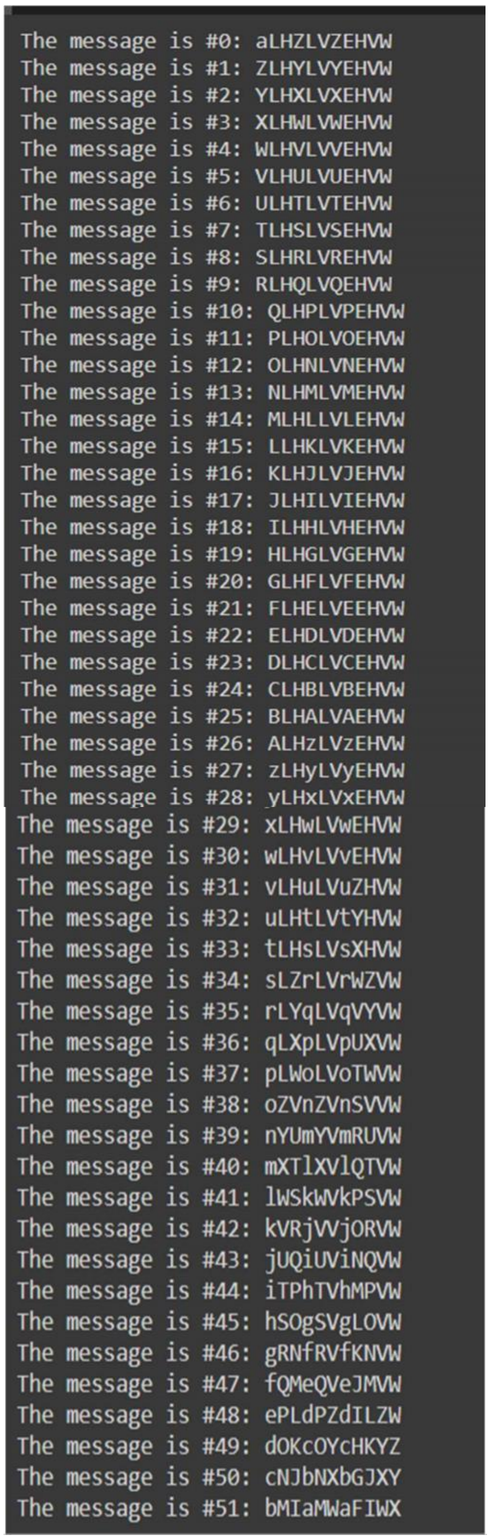
We can hack the Caesar cipher by using a cryptanalytic technique called brute-force. A brute-force attack tries every possible decryption key for a cipher. Decrypting the ciphertext with that key, looking at the output, and then moving on to the next key if they didn’t find the secret message. Because the brute-force technique is so effective against the Caesar cipher.

Algorithm:

Code:

```
cipher_text= aLH LV EHVW
for key in range(len(letters)):
    output = ''
    for symbol in cipher_text:
        num = letters.find(symbol)
        num = num - key
        if num < 0:
            num = num + len(letters)
            output = output + letters[num]
        else:
            output = output + symbol
    print('The message is #%s: %s' %(key, output))
```

Output:



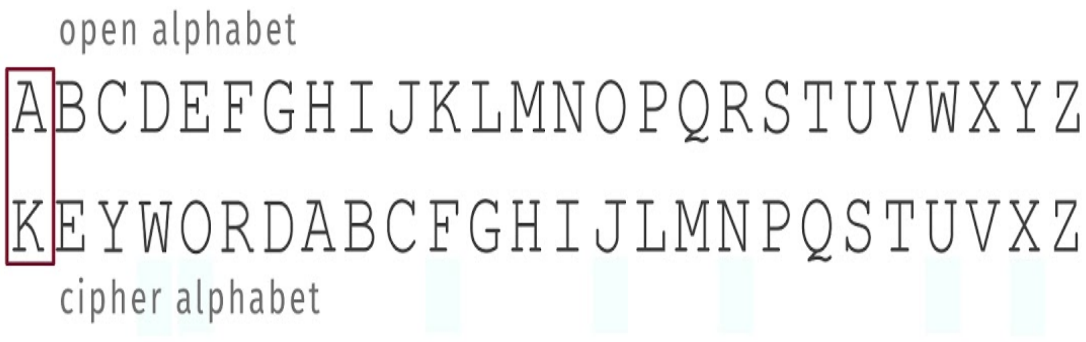
```
The message is #0: aLHZLVZEHVW
The message is #1: ZLHYLVYEHVW
The message is #2: YLHXLVXEHVW
The message is #3: XLHWLVWEHVW
The message is #4: WLHVLVVEHVW
The message is #5: VLHULVUEHVW
The message is #6: ULHTLVTEHVW
The message is #7: TLHSLVSEHVW
The message is #8: SLHRLVREHVW
The message is #9: RLHQLVQEHVW
The message is #10: QLHPLVPEHVW
The message is #11: PLHOLVOEHVW
The message is #12: OLHNLVNEHVW
The message is #13: NLHMLVMEHVW
The message is #14: MLHLLVLEHVW
The message is #15: LLHKLVKEHVW
The message is #16: KLHJLVJEHVW
The message is #17: JLHILVIEHVW
The message is #18: ILHHLVHEHVW
The message is #19: HLHGLVGEHVW
The message is #20: GLHFLVFEHVW
The message is #21: FLHELVEEHVW
The message is #22: ELHDLVDEHVW
The message is #23: DLHCLVCEHVW
The message is #24: CLHBLVBEHVW
The message is #25: BLHALVAEHVW
The message is #26: ALHzLVzEHVW
The message is #27: zLHyLVyEHVW
The message is #28: yLHxLVxEHVW
The message is #29: xLHwLVwEHVW
The message is #30: wLHvLVvEHVW
The message is #31: vLHuLVuZHVW
The message is #32: uLHtLVtYHVW
The message is #33: tLHsLVsXHVW
The message is #34: sLZrLVrWZVW
The message is #35: rLYqLVqVYVW
The message is #36: qLXpLVpUXVW
The message is #37: pLWoLVoTWVW
The message is #38: oZVnZVnSVVW
The message is #39: nYUmYVmRUVW
The message is #40: mXTlXVlQTVW
The message is #41: lWSkWVkPSVW
The message is #42: kVRjVVjORVW
The message is #43: jUQiUViNQVW
The message is #44: iTPhTVhMPVW
The message is #45: hSOgSVgLOVW
The message is #46: gRNfRVfKNVW
The message is #47: fQMeQVeJMVW
The message is #48: ePLdPZdILZW
The message is #49: dOKcOYcHKYZ
The message is #50: cNJbNXbGJXY
The message is #51: bMIaMWaFIWX
```

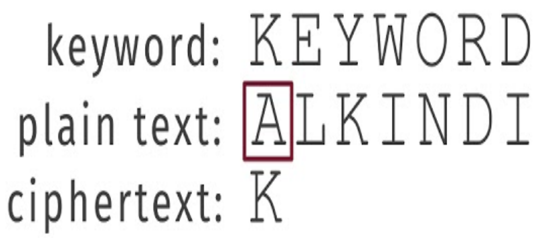
2a] Mono alphabetic Cipher:

Aim:

Theory:

Mono alphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if ‘A’ is encrypted as ‘D’, for any number of occurrence in that plaintext, ‘A’ will always get encrypted to ‘D’.





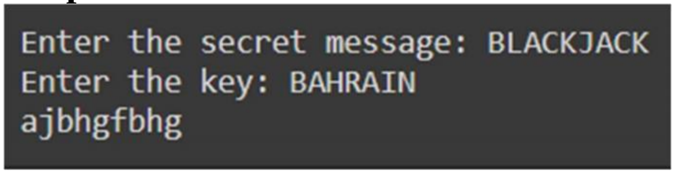
Algorithm:

Code:

#Encryption

```
plain_text = input("Enter the secret message: ").lower()
letters = "abcdefghijklmnopqrstuvwxyz"
key = input("Enter the key: ").lower()
new_key = ""

new_text = ""
cipher_text= []

for char in plain_text:
    if char in letters:
        new_text += char

for char in key:
    if char in letters:
        if char not in new_key:
            new_key += char

for char in letters:
    if char not in new_key:
        new_key += char

def encrypt():
    index_values=[letters.index(char) for char in new_text]
    return "".join(new_key[indexKey] for indexKey in index_values)
print(encrypt())
```

Output:



```
Enter the secret message: BLACKJACK
Enter the key: BAHRAIN
ajbhgfbhg
```

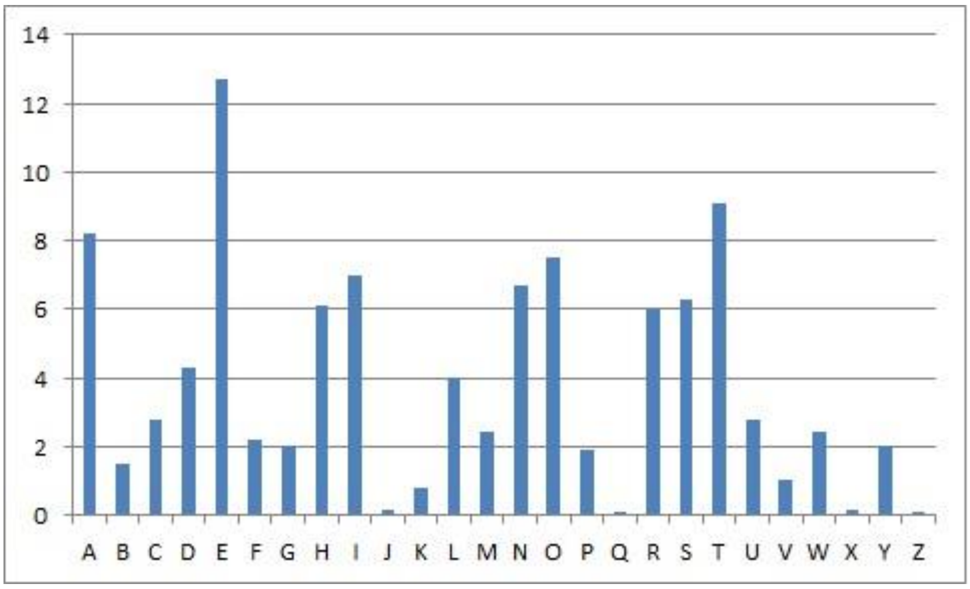
2b] Guessing Attack using Frequency Analysis on Mono Alphabetic Cipher:

Aim:

Theory:

The methodology behind frequency analysis relies on the fact that in any language, each letter has its own personality. The most obvious trait that letters have is the frequency with which they appear in a language. Clearly in English the letter "Z" appears far less frequently than, say, "A". In times gone by, if you wanted to find out the frequencies of letters within a language, you had to find a large piece of text and count each frequency. Now, however, we have computers that can do the hard work for us. But in fact, we don't even need to do this step, as for most languages there are databases of the letter frequencies, which have been calculated by looking at millions of texts, and are thus very highly accurate.

From these databases we find that "E" is the most common letter in English, appearing about 12% of the time (that is just over one in ten letters is an "E"). The next most common letter is "T" at 9%. The full frequency list is given by the graph below.

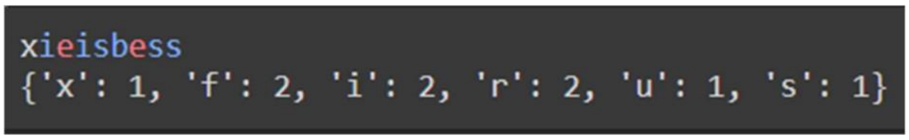


Algorithm:

Code:

```
cipher_text = "xfifruirs"
stored_letters={}

for char in cipher_text:
    if char not in stored_letters:
        stored_letters[char] = 1
    else:
        stored_letters[char] += 1
attempt = cipher_text.replace("i","\033[31me\033[0m")
attempt = attempt.replace("f","\033[34mi\033[0m")
attempt = attempt.replace("r","\033[34ms\033[0m")
attempt = attempt.replace("s","\033[34ms\033[0m")
attempt = attempt.replace("u","\033[34mb\033[0m")
print(attempt)
print(stored_letters)
```

Output:



```
xieisbess
{'x': 1, 'f': 2, 'i': 2, 'r': 2, 'u': 1, 's': 1}
```

Conclusion:

With this experiment we were able to learn and implement encryption and decryption in Caesar Cipher and Monoalphabetic Cipher. The execution of the respective ciphering techniques were also performed with their algorithms.