**PRACTICAL 4**

**Program 1**: Implement polyalphabetic cipher encryption-decryption.

**Code**:

def char\_to\_int(text):

l1 = []

l1.clear()

for char in text:

if char.isalpha():

if char.isupper():

l1.append(ord(char) - 65)

else:

l1.append(ord(char) - 97)

return l1

def int\_to\_chat(number\_list):

l1 = []

for integer in number\_list:

l1.append(chr(integer + 97))

return l1

def key\_generate(text,key):

l1 = []

l2 = []

for i in key:

l2.append(i)

for i in range(len(text)):

j = i % len(l2)

l1.append(l2[j])

return "".join(l1)

def polyalphabetic\_encoding(text, key):

plain\_text\_int = char\_to\_int(text)

key\_text\_int = char\_to\_int(key)

l1 = []

if(len(plain\_text\_int) == len(key\_text\_int)):

for i in range(0, len(plain\_text\_int)):

s1 = plain\_text\_int[i] + key\_text\_int[i]

l1.append(s1)

for i in range(len(l1)):

if(l1[i] > 25):

num = l1[i] - 26

l1[i] = num

encoing = "".join(int\_to\_chat(l1))

return encoing

def polyalphabetic\_decoding(text, key):

decoded\_int = char\_to\_int(text)

key\_decoded\_int = char\_to\_int(key)

l2 = []

if(len(decoded\_int) == len(key\_decoded\_int)):

for i in range(len(decoded\_int)):

s2 = decoded\_int[i] - key\_decoded\_int[i]

l2.append(s2)

for i in range(len(decoded\_int)):

if(l2[i] < 0):

num = l2[i] + 26

l2[i] = num

decoing = "".join(int\_to\_chat(l2))

return decoing

text = input("enter the plain text: ")

key\_text = input("enter the key: ")

key\_generate(text,key\_text)

encoded\_msg = polyalphabetic\_encoding(text,key\_generate(text,key\_text))

decoded\_msg = polyalphabetic\_decoding(encoded\_msg , key\_generate(text,key\_text))

print("encoded massage: ",encoded\_msg)

print("decoded massage: ",decoded\_msg)

**output:**

