**1.**Write a high level code for caesar cipher involves replacing each letter of the alphabet the letter standing at k places further down the alphabet for the k in range 1 to 26

**PROGRAM:**

import java.io.\*;

import java.util.\*;

class Replace{

public static void main(String[] args){

Scanner s=new Scanner(System.in);

String alpha="abcdefghijklmnopqrstuvwxyz";

System.out.println("enter the string:");

String a=s.nextLine();

System.out.println("enter the key value:");

int k=s.nextInt();

a=a.toLowerCase();

String e="";

String d="";

for(int i=0;i<a.length();i++)

{

int p=alpha.indexOf(a.charAt(i));

int ep=(k+p)%26;

char echar=alpha.charAt(ep);

e=e+echar;

}

System.out.println("cipher text"+e);

for(int j=0;j<a.length();j++)

{

int c=alpha.indexOf(e.charAt(j));

int dc=(c-k)%26;

char dchar=alpha.charAt(dc);

d=d+dchar;

}

System.out.println("plain text"+d);

}

}

**OUTPUT:**

enter the string:

class

enter the key value:

5

cipher text:hqfxx

plain text:class

1. **Write a high level code for monoalphabetic cipher maps a plain text alphabet to a cipher letter alphabet so that each letter of the plain text alphabet map to a single unique letter of cipher text alphabet.**

**Program:**

import java.io.\*;

import java.util.\*;

class Monoalpha{

public static void main(String[] args){

Scanner s=new Scanner(System.in);

String alpha="abcdefghijklmnopqrstuvwxyz";

String mono="defghijklmnopqrstuvwxyzabc";

System.out.println("enter the string");

String a=s.nextLine();

a=a.toLowerCase();

String e="";

for(int i=0;i<a.length();i++)

{

int p=alpha.indexOf(a.charAt(i));

char echar=mono.charAt(p);

e=e+echar;

}

System.out.println("cipher text:"+e);

}

}

**Ouput:**

enter the stringhello

cipher text:khoor

1. **Write a high level code for play fair algorithm is based on use of 5x5 matrix of a letter constructed using a keyword plain text is encrypted two letters at a time using this matrix**.

**Program:**

def create\_matrix(key):

key = key.upper().replace("J", "I") + "ABCDEFGHIKLMNOPQRSTUVWXYZ"

matrix = []

for char in key:

if char not in matrix:

matrix.append(char)

return matrix

def playfair\_cipher(key, text):

matrix = create\_matrix(key)

text = text.upper().replace("J", "I").replace(" ", "")

cipher = []

for i in range(0, len(text), 2):

c1, c2 = text[i], text[i + 1] if i + 1 < len(text) else 'X'

r1, c1 = divmod(matrix.index(c1), 5)

r2, c2 = divmod(matrix.index(c2), 5)

if r1 == r2:

cipher.append(matrix[r1 \* 5 + (c1 + 1) % 5])

cipher.append(matrix[r2 \* 5 + (c2 + 1) % 5])

elif c1 == c2:

cipher.append(matrix[(r1 + 1) % 5 \* 5 + c1])

cipher.append(matrix[(r2 + 1) % 5 \* 5 + c2])

else:

cipher.append(matrix[r1 \* 5 + c2])

cipher.append(matrix[r2 \* 5 + c1])

return ''.join(cipher)

key = input("Keyword: ")

text = input("Text: ")

print("Cipher: " + playfair\_cipher(key, text))

**Output:**

Keyword: APPLE

Text: HELLO

Cipher: GBEEMZ

1. **Write a high level code for poly alphabetic substitution cipher uses a separate mono alphabetic cipher for each succesive letter of plain text depending on a key.**

**Program:**

def polyalphabetic\_cipher(text, key):

encrypted\_text = ""

key\_length = len(key)

for i in range(len(text)):

char = text[i]

if char.isalpha():

shift = ord(key[i % key\_length].lower()) - ord('a')

if char.isupper():

encrypted\_char = chr(((ord(char) - ord('A') + shift) % 26) + ord('A'))

else:

encrypted\_char = chr(((ord(char) - ord('a') + shift) % 26) + ord('a'))

else:

encrypted\_char = char

encrypted\_text += encrypted\_char

return encrypted\_text

message = input("Enter a message: ")

keyword = input("Enter a keyword: ")

encrypted\_message = polyalphabetic\_cipher(message, keyword)

print("Encrypted message:", encrypted\_message)

**OUTPUT:**

Enter a message: HELLO

Enter a keyword: BRO

Encrypted message: IVZMF

1. **Write a high level code for cipher text has been generated affiend cipher the most frequent letter of the cipher text is “B” and second most frequent letter of cipher text is “u” break this code.**

Program:

def break\_affine\_cipher(c, a, b):

return ''.join([chr((a \* (ord(x) - 65 - b) % 26) + 97) if x.isalpha() else x for x in c])

ciphertext = input("Enter the ciphertext: ").upper()

most\_common = 'B'

second\_most\_common = 'U'

a = (ord(most\_common) - ord(second\_most\_common)) % 26

b = (ord(most\_common) - ord('A')) % 26

plaintext = break\_affine\_cipher(ciphertext, a, b)

print(plaintext)

Output:

Enter the ciphertext: hello

Qvssn

6.Write a C program forgeneralization of the Caesar cipher, known as the affine Caesar cipher, has thefollowing form: For each plaintext letter p, substitute the ciphertextletterC: C = E([a, b], p) = (ap + b) mod 26 A basic requirement of any encryption algorithm is that it be one-to-one. That is, if p q,then E(k, p) E(k, q). Otherwise, decryption is impossible, because more than one plaintext character maps into the same ciphertext character. The affine Caesar cipher is not one-to one for all values of a. For example, for a = 2 and b = 3,then E([a,b], 0) = E([a, b], 13) = 3.

a. Are there any limitations on the value of b?

b. Determine which values of a are not allowed.

Program:

def affine\_caesar(text, a, b, encrypt=True):

result = ""

for char in text:

if char.isalpha():

shift = ord('A') if char.isupper() else ord('a')

char\_num = ord(char) - shift

if encrypt:

new\_char\_num = (a \* char\_num + b) % 26

else:

a\_inverse = None

for i in range(26):

if (a \* i) % 26 == 1:

a\_inverse = i

break

if a\_inverse is None:

return "Decryption not possible with this 'a' value."

new\_char\_num = (a\_inverse \* (char\_num - b)) % 26

result += chr(new\_char\_num + shift)

else:

result += char

return result

plaintext = input("Enter the text: ")

a = int(input("Enter the A value: "))

b = int(input("Enter the B value: "))

encrypted\_text = affine\_caesar(plaintext, a, b, encrypt=True)

print("Encrypted:", encrypted\_text)

decrypted\_text = affine\_caesar(encrypted\_text, a, b, encrypt=False)

print("Decrypted:", decrypted\_text)

Output:

Enter the text: hello

Enter the A value: 15

Enter the B value: 5

Encrypted: gnooh

Decrypted: hello