Page No. ______
Date

① CAESAR - CIPHER

- Formulae :-

$$E_n(x) = (x+n) \bmod 26$$
(Encryption Phase with shift n)
$$D_n(x) = (x-n) \bmod 26$$
(Decryption Phase with shift n)

- Algorithm for Caesar Cipher

I/P : ① A string of lower case letters, called text.
② An Integer between 0-25 denoting the required shift.

Procedure : ① Traverse the given text one character at a time.
② For each character, transform the given character as per the rule, depending on wether we're encrypting or decrypting text.
③ Return the new string generated.

Cipher (n) = Decipher $(26-n)$

Page No.
Date

② VIGERE CIPHERE

- Encryption:- In order to cipher a tent, take the first letter of the message and the first letter of the key, add their value (letters have a value depending on their rank in the alphabet, starting with 0). The result of the addition modulo 26 gives the rank of the ciphered letter.
Do the same for each character in the text.

- Decryption :- To decrypt, take the first letter of the ciphertext and the first letter of the key, and subtract their value (letters have a value equal to their position in the alphabet starting from 0). If the result is negative, add 26, the result gives the rank of the plain letter.
Do the same for each character in the cipher text.

To find the key ① find key length using Kaisiki test.
② use frequency analysis chi-square test to find each letter of the key separately.

$E_i = (T_i + K_i) \bmod 26$
(Encryption Phase for every ith character)
$D_i = (E_i - K_i + 26) \bmod 26$
(Decryption Phase for every ith character)

Page No.
Date

③ PLAYFAIR CIPHER

Encryption:-

① generate the key square (6×6):

- It is a key square is 6×6 grid of alphabets that acts a key for encrypting the plaintext. Each of the 36 characters must be unique and 10 characters after the 26 english alphabet letters are numbers from [0, 9].
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

② Process to encrypt the plain-text: The text is split into pair of two letters (digraphs). If there is an odd number of letters, a 'z' is added to the last letter.

- If both the ~~letters~~ characters are in the same column: Take the character below each one (going back to the top if at the bottom).
- If both the characters are in the same row: Take the right of each One (going back to the leftmost if at the rightmost).
- If neither of the above rules is true: Form a rectangle with the two letters and take the letters On the horizontal opposite corner of the rectangle.

Decryption:-

Process to decrypt the cipher-text: The text is split

Page No.
Date

into pair of two letters (digraphs). If there is an odd number of letters a 'Z' is added to the last letter.

- If both the letters are in the same column: Take the character above each one (going back to the ~~top~~ bottom if at the ~~bot~~ top).
- If both the letters are in the same row :- Take the character to the left of each one (going back to the rightmost if at the leftmost position).
- If neither of the above rules is true: Form a rectangle with the two characters and take the characters on the horizontal opposite corner of the rectangle.

Page No.
Date

Modular Inverse using Extended Euclidian :-

```
#include <iostream>
using namespace std;

int arr [10] = {0, 1};

void modulusExtended (int arr[], int K , int n) {
        if (arr [K]  >= 0) {
                arr [K] = arr [K] % n;
        } else {
                arr [K]  =  n + arr [K];
        }
}
void clearExt (int arr []) {
        for (int i = 2; i < 10; i++) {
                arr [i] = 0;
        }
}

int  extended (int a , int n) {
     clear Ext (arr);
     int N = n;
     int K = 2;
     while ( n % a != 0) {
          int q = n/a;
          arr [K] = arr [K-2] - (arr [K-1] * q);
          modulusExtended (arr, K, N);
          K++;
          int temp = a;
          a = n % a;
     }    n = temp;
```

Page No.

Date

```
int  main() {
        int a, b;
        cin >> a >> b;
        cout << extended (a, b) << endl;
        return 0 ;
  }
```