NIS

LAB - 2

ROll NO. : CE146
Name : Shingala Shubham P.
ID NO. : 19CEUOS159

* **AIM** : Write a program to implement
1) Extended Euclidean Algorithm for finding multiplicative inverse
2) Multiplicative cipher
3) Affine cipher.

1) Extended Euclidean Algorithm

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

int Extended_Euclidean ( int a , int b) {
    int q, r, t, t1=0 , t2=1 , r1=b , r2=a;
    while (r2 >0) {
            q= r1/r2 ;
            r= r1 - q*r2 ;
            r1 = r2 ;
            r2 = r ;
            t= t1 - q*t2 ;
            t1 = t2 ;
            t2= t ;

    }
if (r1 = = 1) {
        if (t1 < 0)
            t1 += b;
        return t1;
    }
    return 0;
}

int main() {
    int a, b;
    cout << " Enter two numbers: " << endl;
    cin >> a >> b;
    if ( int ans = Extended_Eucliden (a, b)) {
        cout << " multipicative inverse
                is " << ans;
    }
    else {
        cout << "multipiicative inverse is
                not possible";
    }
    return 0;
}
```

→ Test-case - 1 : Input : 2 4
output : multiplicative inverse is not possible

→ Test-case - 2 : Input : 5 21
output : multiplicative inverse is 17

→ Test-case - 3 : Input : 11 7
output : multiplicative inverse is 2

2) multiplicative cipher

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

bool gcd (int key , int number) {
    while (key > 0) {
            int temp = key;
            key = number % key
            number = temp;
    }
    if (number = = 1)
        return true;
    return false;
}

int multiplicativeInverse (int a, int b) {
    // it is same as previous program's
        Extended _ Euclidean function
}
```

```
string   encrypt (string plainText, int key) {
     string  encryptText ;
     int  length = plainText.length() ,
               i, textmap ;
     for ( i=0 ; i<length ; i++ ) {
          if(isupper (plainText[i]))
               textMap = plainText[i] - 'A' ;
          else
               textmap = plainText[i] - 'a' ;
          encryptText += (textmap * key)%26
                              + 'A' ;
     }
     return  encryptText ;
 }

 string  decrypt (string encryptText , int key) {
      int Inversekey = multipilicative_inverse(
                         key, 26) ;
      string   decryptText ;
      for (int i=0 ; i<encryptText.length() ; i++) {
           int textmap  =  encryptText[i] - 'A' ;
           textmap =  (textmap * Inversekey)%26;
           if (textmap <0)
                textmap += 26 ;
           decryptText + = textmap + 'a' ;
      }
      return  decryptText ;
 }
```

```
int   main ()   {
          string   plainText ;  int   key ;
          cout << " Enter   Plain   Text :  " ;
          cin >>  plainText ;
          cout << " Enter   key : " ;
          cin >>  key ;
          while ( ! gcd( key, 26)) {
                 cout << " Enter key  again : " ;
                 cin >>  key ;
          }
          string   encrypText = encrypt(plainText,
                                          key ) ;
          cout << " Encrypt Text :" << encrypText
           << endl << " Decrypt  Text : " <<
              decrypt ( encryptText, key ) ;
}
```

→ Test case - 1 :

Input : Shubham Shingala   Key = 7

output : encrypt text : WXKHXAGNX ENQAZA

decrypt text : Shubham Shingala

→ Test case - 2 :

Input : hello   Key = 4

output : Enter key again : 9

→ encrypt text : LKVVW

decrypt Text : Hello

3) Affine cipher

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

bool gcd (int key, int number) // is same as
    previous code
int multipilcartive inverse (int a, int b)
    // is same as previous code

string encrypt ( string plainText, int key1,
                    int key2) {
    string encrypt Text ;
    for ( int i=0 ; i< plainText.length() ; i++) {
        int textmap = PlainText [i] ;
        if (isupper (textmap))
              textmap -= 'A' ;
        else
              textmap - = 'a' ;
        encrypt Text += ((textmap * key1)
                   +key2) % 26 + 'A' ;
    }
    return encrypt Text,
  }
```

```
string  decrypt(string  encryptText, int key1
              , int  key2) {
     string  decryptText;
     int  InverseKey1 = multipiicative_inverse
                        (key1, 26);
     for(int i=0; i< encryptText.length(); i++){
          int  textmap = encryptText[i]-'A';
          textmap = ((textmap - key2) *
              Inverse key1) % 26  ;
          if( textmap <0)
              textmap+ = 26;
          decryptText + = textmap + 'a';
     }
     return  decrypt Text;
}

int  main() {
     string  PainText;
     cout << " Enter  plain  Text: \n ";
     cin >>  PlainText;
     int  key1, key2;
     cout << " Enter  key 1 :" ;
     cin >> key1
     cout << " Enter  key2 :  " ;
     cin >>  key 2
     string  encryptText = encrypt(PlainText,
                    key1, key2) ;
```

```
        cout << " Encrypt Text : " << encryptText
            << endl << " DecryptText " <<
             decrypt (encryptText , key1 , key2);
        return 0;
    }
```

→ Test case -1 :

Input : shubham shingala   key1 = 7   key2 = 12

output : Encrypt Text : IJWTJMSIJQZCMLM
              Decrypt Text : shubham shingala

→ Test case -2 :

Input : affine cipher   key 1 = 17   key2 = 9

output : Encrypt Text : JQQ PWZRPEYZM
              Decrypt Text : affinecipher.