a) caesar cipher

```
#include "stdc++.h"
using namespace std;


void plainToEncrypt(){
	int k;
	cout<<"Enter K value : ";
	cin>>k;
	string plainText;
	cout<<"Enter Plain text :";
	cin>>plainText;
	string encrypt = "";
	for(auto x : plainText){
	if(x >= 'a' && x <= 'z'){
	int val = x - 'a';
	val += k;
	int sub = val%26;
	encrypt += (sub + 'a');
	}else{
	int val = x - 'A';
	val += k;
	int sub = val%26;
	encrypt += (sub + 'A');
	}
	}

	cout<<"Encrypted Text is :"<<encrypt<<endl;
}
void EncryptToPlain(){
	int k;
	cout<<"Enter K value : ";
	cin>>k;
	string encrypt;
	cout<<"Enter Encrypt text : ";
	cin>>encrypt;
	string plainText = "";
	for(auto x : encrypt){
	if(x >= 'a' && x <= 'z'){
	int val = x - 'a';
	val -= k;
	int sub = (val+26)%26;
	plainText += (sub + 'a');
	}else{
	int val = x - 'A';
	val -= k;
	int sub = (val+26)%26;
	plainText += (sub + 'A');
	}
	}

	cout<<"Plain Text is : "<<plainText<<endl;
}
int main() {

	int choice;
	while(true){
	cout<<"1. Encrypt Text"<<endl<<"2. Decrypt Text"<<endl<<"3. End"<<endl;
	cin>>choice;
	if(choice == 1)
	{
	plainToEncrypt();
	}else if(choice == 2){
	EncryptToPlain();
	}else{
	break;
	}
```

```
            cout<<endl<<endl;
            }
            return 0;
}
```

b) Playfair cipher

```
#include<bits/stdc++.h>
using namespace std;
int main() {
    string key;
    cin>>key;
    map<char , pair<int , int>> chartopair;
    map<pair<int , int> , char> pairtochar;
    vector<vector<char>> mat(5 , vector<char>(5));
    int ptrrow = 0;
    int ptrcol = 0;
    for(auto x : key){
        if(x == 'j'){
            x = 'i';
        }
        if(chartopair.find(x) != chartopair.end()) continue;
        chartopair[x] = {ptrrow , ptrcol};
        mat[ptrrow][ptrcol] = x;

        ptrcol++;
        if(ptrcol == 5){
            ptrcol = 0;
            ptrrow++;
        }
    }
    for (int i = 0; i < 26; ++i)
    {
        if(i + 'a' == 'j'){
            continue;
        }
        if(chartopair.find(i + 'a') != chartopair.end()) continue;
        chartopair[i+'a'] = {ptrrow , ptrcol};
        mat[ptrrow][ptrcol] = i+'a';
        ptrcol++;
        if(ptrcol == 5){
            ptrcol = 0;
            ptrrow++;
        }
    }

    for(auto x : mat){
        for(auto y : x){
            cout<<y<<" ";
        }
        cout<<endl;
    }

    string s;
    cin>>s;
    string plain = "";
    plain += s[0];

    for(int i = 1 ; i < s.length(); i++){
        if(s[i] == s[i-1]){
            plain += 'x';
        }
```

```
        plain += s[i];
    }

    if(plain.length()&1){
        plain += 'x';
    }
    string enc = "";

    for (int i = 0; i < plain.length(); i+=2)
    {
        char first = plain[i];
        char second = plain[i+1];
        int firstrow = chartopair[first].first;
        int firstcol = chartopair[first].second;
        int secondrow = chartopair[second].first;
        int secondcol = chartopair[second].second;

        enc += mat[firstrow][secondcol];
        enc += mat[secondrow][firstcol];

    }
    cout<<enc<<endl;


    string dec = "";

    for (int i = 0; i < enc.length(); i+=2)
    {
        char first = enc[i];
        char second = enc[i+1];
        int firstrow = chartopair[first].first;
        int firstcol = chartopair[first].second;
        int secondrow = chartopair[second].first;
        int secondcol = chartopair[second].second;

        dec += mat[firstrow][secondcol];
        dec += mat[secondrow][firstcol];

    }
    cout<<dec;

    return 0;
}
```

c) Hill cipher

```
#include<bits/stdc++.h>
using namespace std;

int main() {
    vector<vector<int>> v(3 , vector<int>(3));
    v = {{6 , 24 , 1},
         {13 , 16 , 10},
         {20 , 17 , 15}};
    vector<int> plain;
    string sp;
    cin>>sp;
    vector<int> enc;
    for(auto x : sp){
        plain.push_back(x-'a');
    }
    for (int i = 0; i < 3; ++i)
    {
        int temp = 0;
```

```
        for (int j = 0; j < 3; ++j)
        {
            temp += v[i][j]*plain[j];
        }
        enc.push_back(temp%26);
    }
    for(auto x : enc) cout<<(char)(x + 'a');
    return 0;
}
```

d) Vigenere cipher

```
#include<bits/stdc++.h>
using namespace std;
//yash dhanlobhe
int main(){

 cout<<"Vigenere Cipher Encryption\n";
 int i,j,k,n;
 vector<vector<char> > a(26,vector<char>(26));
 k=0;
 n=26;
 for(i=0;i<n;i++){
 k=i;
 for(j=0;j<n;j++){
 a[i][j]='A'+k;
 k++;
 if(k==26)
 k=0;
 }
 }
 cout<<"Enter the message\n";
 string s;
 cin>>s;
 cout<<"Enter the key\n";
 string key;
 cin>>key;
 k=0;
 int mod = key.size();
 for(i=key.size();i<s.size();i++){
 key+=key[k%mod];
 k++;
 }
 string encrypt;
 for(i=0;i<s.size();i++){
 encrypt+= a[s[i]-'A'][key[i]-'A']; }
 cout<<"Encrypted message: "<<encrypt<<'\n';
 return 0; }
#include<bits/stdc++.h>
using namespace std;
//yash dhanlobhe
int main(){
 cout<<"Vigenere Cipher Decryption\n";
 int i,j,k,n;
 vector<vector<char> > a(26,vector<char>(26));
 k=0;
 n=26;
 for(i=0;i<n;i++){
 k=i;
 for(j=0;j<n;j++){
 a[i][j]='A'+k;
 k++;
 if(k==26) k=0; } }
 cout<<"Enter the encrypted message\n";
```

```
string s;
cin>>s;
cout<<"Enter the key\n";
string key;
cin>>key;
k=0;
for(i=key.size();i<s.size();i++){
key+=key[k];
k++;
}
string decrypt;
for(i=0;i<s.size();i++){
for(j=0;j<n;j++){
if(a[j][key[i]-'A']==s[i]){
decrypt += 'A'+j;
break; } } }
cout<<"Decrypted message: "<<decrypt<<'\n';
return 0; }
```

- Row columnar

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    string s;
    cin>>s;
    string ns = "";
    set<char> st;
    for(auto x : s){
        if(st.find(x) == st.end()){
            st.insert(x);
            ns += x;
        }
    }
    vector<vector<int>> v;
    for (int i = 0; i < ns.length(); ++i)
    {
        v.push_back({ns[i] - '0' , i});
    }
    sort(v.begin() , v.end());

    char arr[500][500] = {'#'};
    int maxx = ns.length();
    int row = 0;
    int col = 0;
    string pt;
    cin>>pt;
    for(auto x : pt){
        arr[row][col] = x;
        col++;
        if(col == maxx){
            col = 0;
            row++;
        }
    }
    string enc = "";
    for(auto x : v){
        for(int i = 0 ; i <= row  ; i++){
            enc += arr[i][x[1]];
        }
```

```
    }
    cout<<enc;
    return 0;
}
```

- Rail fence

```
#include <bits/stdc++.h>
//yash dhanlobhe
using namespace std;

string encryptRailFence(string text, int key)
{

    char rail[key][(text.length())];

    for (int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] = '\n';

    bool dir_down = false;
    int row = 0, col = 0;

    for (int i=0; i < text.length(); i++)
    {
        if (row == 0 || row == key-1)
            dir_down = !dir_down;
        rail[row][col++] = text[i];
        dir_down?row++ : row--;
    }
    string result;
    for (int i=0; i < key; i++)
        for (int j=0; j < text.length(); j++)
            if (rail[i][j]!='\n')
                result.push_back(rail[i][j]);

    return result;
}

string decryptRailFence(string cipher, int key)
{
    char rail[key][cipher.length()];

    for (int i=0; i < key; i++)
        for (int j=0; j < cipher.length(); j++)
            rail[i][j] = '\n';
    bool dir_down;
    int row = 0, col = 0;
    for (int i=0; i < cipher.length(); i++)
    {
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        rail[row][col++] = '*';

        dir_down?row++ : row--;
    }
    int index = 0;
    for (int i=0; i<key; i++)
```

```
        for (int j=0; j<cipher.length(); j++)
            if (rail[i][j] == '*' && index<cipher.length())
                rail[i][j] = cipher[index++]
    string result;
    row = 0, col = 0;
    for (int i=0; i< cipher.length(); i++)
    {
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        if (rail[row][col] != '*')
            result.push_back(rail[row][col++]);

        dir_down?row++: row--;
    }
    return result;
}
void solve(){
    cout <<"Encryption of \"YASH DHANLOBHE\":- "<< encryptRailFence("YASH DHANLOBHE", 2) <<
endl;
    cout << "Decryption of \"YS HNOHAHDALBE\":- "<<decryptRailFence("YS HNOHAHDALBE",2) <<
endl;
   }

int main() {
   solve();
   return 0;
}
```

- Implement Data Encryption Standard

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    cout<<"Enter 10 bit key: ";
    int key[10];
    int i;
    for(i=0;i<10;i++){
        cin>>key[i];
    }
    int p10[]={3,5,2,7,4,10,1,9,8,6};
    int p8[]={6,3,7,4,8,5,10,9};
    int knew[10];
    //appyling p10
    for(i=0;i<10;i++){
        knew[i]=key[p10[i]-1];
    }
    //left shift
    int a=knew[0],b=knew[5];
    for(i=0;i<10;i++){
        if(i==4)
            i++;
        knew[i]=knew[i+1];
    }
    knew[4]=a;knew[9]=b;
    //applying p8
    int temp[8];
    for(i=0;i<8;i++){
        temp[i]=knew[p8[i]-1];
    }
```

```
    cout<<"key k1: ";
    for(i=0;i<8;i++){
        cout<<temp[i];
    }

    //again left shift
    int x=knew[0],y=knew[5];
    for(i=0;i<10;i++){
        if(i==4)
            i++;
        knew[i]=knew[i+1];
    }
    knew[4]=x;knew[9]=y;

    //applying p8
    int temp2[8];
    for(i=0;i<8;i++){
        temp2[i]=knew[p8[i]-1];
    }
    cout<<"key k2: ";
    for(i=0;i<8;i++){
        cout<<temp2[i];
    }

    return 0;
}

int[] function_(int[] ar, int[] key_)
    {

        int[] l = new int[4];
        int[] r = new int[4];

        for (int i = 0; i < 4; i++) {
            l[i] = ar[i];
            r[i] = ar[i + 4];
        }

        int[] ep = new int[8];

        for (int i = 0; i < 8; i++) {
            ep[i] = r[EP[i] - 1];
        }

        for (int i = 0; i < 8; i++) {
            ar[i] = key_[i] ^ ep[i];
        }

        int[] l_1 = new int[4];
        int[] r_1 = new int[4];

        for (int i = 0; i < 4; i++) {
            l_1[i] = ar[i];
            r_1[i] = ar[i + 4];
        }

        int row, col, val;

        row = Integer.parseInt("" + l_1[0] + l_1[3], 2);
        col = Integer.parseInt("" + l_1[1] + l_1[2], 2);
        val = S0[row][col];
        String str_l = binary_(val);

        row = Integer.parseInt("" + r_1[0] + r_1[3], 2);
```

```
        col = Integer.parseInt("" + r_1[1] + r_1[2], 2);
        val = S1[row][col];
        String str_r = binary_(val);

        int[] r_ = new int[4];
        for (int i = 0; i < 2; i++) {
            char c1 = str_l.charAt(i);
            char c2 = str_r.charAt(i);
            r_[i] = Character.getNumericValue(c1);
            r_[i + 2] = Character.getNumericValue(c2);
        }
        int[] r_p4 = new int[4];
        for (int i = 0; i < 4; i++) {
            r_p4[i] = r_[P4[i] - 1];
        }

        for (int i = 0; i < 4; i++) {
            l[i] = l[i] ^ r_p4[i];
        }

        int[] output = new int[8];
        for (int i = 0; i < 4; i++) {
            output[i] = l[i];
            output[i + 4] = r[i];
        }
        return output;
    }
```

- Implement Advance Encryption Standard

```
import java.security.SecureRandom;
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.spec.SecretKeySpec;
import java.util.Random;
import java.util.Scanner;

class AES {
    static byte[] raw;
    static String inputMessage;

    static void generateSymmetricKey() {
        try {
            Random r = new Random();
            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
            sr.setSeed(String.valueOf(r.nextInt(10000)).getBytes());
            kgen.init(256, sr);
            raw = kgen.generateKey().getEncoded();
            System.out.println("DES Symmetric key = "+raw.toString());
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }


    private static byte[] encrypt(byte[] raw, byte[] clear) throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw,"AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec);
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }
```

```
    private static byte[] decrypt(byte[] raw, byte[] encrypted)throws Exception
    {
        SecretKeySpec skeySpec = new SecretKeySpec(raw,"AES");
        Cipher cipher = Cipher.getInstance("AES/ECB/PKCS5PADDING");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec);
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }
    public static void main(String args[]) {
        Scanner sc  = new Scanner(System.in);
        try
        {
            generateSymmetricKey();
            System.out.println("Enter message to encrypt");
            inputMessage= sc.nextLine();
            byte[] ibyte = inputMessage.getBytes();
            byte[] ebyte=encrypt(raw, ibyte);
            String encryptedData = new String(ebyte);
            System.out.println("Encrypted message "+encryptedData);
            byte[] dbyte= decrypt(raw,ebyte);
            String decryptedMessage = new String(dbyte);
            System.out.println("Decrypted message"+decryptedMessage);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

- Implement Diffie Hellman Key exchange algorithm

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    int mod , pb , privatea , privateb;
    cin>>mod>>pb>>privatea>>privateb;

    int asend = pow(pb , privatea);
    asend = asend%mod;
    int bsend = pow(pb , privateb);
    bsend = bsend%mod;

    int bready = pow(asend , privateb);
    bready = bready%mod;
    int aready = pow(bsend , privatea);
    aready = aready%mod;

    cout<<bready<<aready;

    return 0;
}
```

- Implement RSA algorithm

```
#include<bits/stdc++.h>

using namespace std;

int main() {
```

```
    double  p = 3 , q = 7;
    double n = p*q;
    double totient = (p-1)*(q-1);
    double e = 2;
    while(true){
        if(__gcd((int)e , (int)totient) == 1) break;
        e++;
    }
    double k = 2;
    double d = (1 + k*totient)/e;
    int msg = 12;

    double cipher = (pow(msg , e));
    // cipher = cipher%n;.;
    cipher = fmod(cipher , n);
    double dcipher = (pow(cipher , d));
    dcipher = fmod(dcipher , n);
    cout<<msg<<dcipher;

    return 0;
}
```

- Implement and write advantages of Poly-alphabetic Cipher.

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    vector<vector<char>> v(26 , vector<char>(26));
    for (int i = 0; i < 26; ++i)
    {
        v[0][i] = 'a' + i;
    }
    for (int i = 1; i < 26; ++i)
    {
        v[i] = v[i-1];
        rotate(v[i].begin(), v[i].begin()+1, v[i].end());
    }
    string plainText = "";
    string key = "";
    string encryptText = "";
    cout<<"Enter Text"<<" ";
    cin>>plainText;
    cout<<"Enter Key"<<" ";
    cin>>key;
    for(int i = 0 ; i < plainText.length() ; i++){
        encryptText += v[key[i%key.length()]-'a'][plainText[i]-'a'];
    }
    cout<<"Encrypt Text "<<encryptText<<endl;
    string decryptText = "";
    for (int i = 0; i < encryptText.length(); ++i)
    {
        int idx = 0;
        while(v[key[i%key.length()]][idx] != encryptText[i]) idx++;
        decryptText += 'a' + idx;
    }
    cout<<"Decrpt Text "<<decryptText<<endl;
    return 0;
}
```

- Implement SHA algorithm.

```
import java.math.BigInteger;
import java.security.*;
```

```
public class YASH {
    public static void main(String args[])
    {
        String s1 = "yashdhanlobhe";
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");
            byte[] messageDigest = md.digest(s1.getBytes());
            BigInteger no = new BigInteger(1, messageDigest);
            String hashtext = no.toString(16);
            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }
            System.out.println(hashtext);
        }
        catch (Exception e) {
        }
    }
}
```

- Implement digital signature standard.

```
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.Signature;
import java.util.Scanner;

public class CreatingDigitalSignature {
   public static void main(String args[]) throws Exception {

      Scanner sc = new Scanner(System.in);
      System.out.println("Enter some text");
      String msg = sc.nextLine();

      KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("DSA");

      keyPairGen.initialize(2048);

      KeyPair pair = keyPairGen.generateKeyPair();

      PrivateKey privKey = pair.getPrivate();

      Signature sign = Signature.getInstance("SHA256withDSA");
      sign.initSign(privKey);
      byte[] bytes = msg.getBytes();

      sign.update(bytes);

      byte[] signature = sign.sign();

      System.out.println("Digital signature for given text: ");
      System.out.println(new String(signature, "UTF8"));
   }
}
```