XORed

```
#include<stdio.h>
void main(){
            char input[]="helloworld";
             char result[sizeof(input)];
             for(int i=0;i<sizeof(input)-1;i++)
                        result[i]=input[i]^0;
             printf("Original String: %s\n",input);
             printf("XORed String: %s\n",result);
}
```

XORed a string 127

```
#include<stdio.h>
void main(){
            char input[]="helloworld";
            char xorresult[sizeof(input)];
            char andresult[sizeof(input)];
            for(int i=0;i<sizeof(input)-1;i++){
                        xorresult[i]=input[i]^127;
                        andresult[i]=input[i]&127;
            }
            printf("Original String: %s\n",input);
            printf("XORed String: %s\n",xorresult);
            printf("ANDed String: %s\n",andresult);
}
```

Caeser Cipher

```
#include<stdio.h>
void main(){
            char input[]="helloworld";
            int i,len=sizeof(input)-1;
            char enc[len],dec[len];
            for(i=0;i<len;i++){
                        enc[i]=(((input[i]-'a')+3)%26)+'a';
                        dec[i]=(((enc[i]-'a')-3)%26)+'a';
            }
            printf("Original string: %s\n", input);
            printf("Encrypted string=%s\t\n",enc);
            printf("Decrypted string=%s\t\n",dec);
}
```

Substitution cipher

```
#include<stdio.h>
void main(){
        Char input[]=”hello”;
        int i,j,len=sizeof(input)-1,index;
        char t[]=”qwertyuioplkjhgfdsazxcvbnm”;
        char enc[len],dec[len];
        for(i=0;i<len;i++){
                index=input[i]-‘a’;
                enc[i]=t[index];
        }
        printf(“Original string:%s\n”, input);
        printf(“Encrypted string=%s\t\n”,enc);
        for(i=0;i<len;i++){
                for(j=0;j<26;j++){
                        if(enc[i]==t[j])
                                dec[i]=j+’a’;
                }
        }
        printf(“Decrypted string=%s\t\n”,dec);
}
```

Hill cipher

```
#include<stdio.h>
void main()
{
        int i,j;
        int key={{2,3},{3,6}};
        char a[]=”attack”;
        int len=sizeof(a)-1;
        char e[len], num[len],d[len];
        for(i=0;i<len;i++){
                num[i]=a[i]-‘a’;
         }
        for(i=0;i<=len;i=i+2){
                e[i]=((num[i]*key)%26+(num[i+1]*key[1])%26)%26;
                e[i+1]=((num[i]*key[1])%26+(num[i+1]*key[1][1])%26)%26;
        }
        char enc[len];
        for(i=0;i<len;i++){
                enc[i]=e[i]+’a’;
        }
        printf(“Original string:%s\n”,a);
        printf(“Encrypted string=%s\t\n”,enc);
        int del= (key*key[1][1])-(key[1]*key[1]);
        int del_inv;
        for(i=0;i<26;i++){
                if((del*i)% 26==1){
                        del_inv=i;
```

```
            break;
        }
    }
    int k_adj={{key[1][1],0-key[1]},{0-key[1],key}};
    int k_inv;
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            k_inv[i][j]=k_adj[i][j]*del_inv;
        }
    }
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            if(k_inv[i][j]<0){
                k_inv[i][j]+=26;
            }
        }
    }
    for(i=0;i<=len;i=i+2){
        d[i]=((e[i]*k_inv)%26+(e[i+1]*k_inv[1])%26)%26;
        d[i+1]=((e[i]*k_inv[1])%26+(e[i+1]*k_inv[1][1])%26)%26;
    }
    char dec[len];
    for(i=0;i<len;i++){
        dec[i]=d[i]+’a’;
    }
    printf(“Decrypted string=%s\t\n”,dec);
}
```

DES

```
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.DESKeySpec;
import java.util.Base64;
public class Des{
     public static void main(String[] args) throws Exception {
        String originalText = “Hello World”;
        String keyString = “01234567”;
        DESKeySpec desKeySpec = new DESKeySpec(keyString.getBytes());
        SecretKey secretKey =SecretKeyFactory.getInstance(“DES”).generateSecret(desKeySpec);
        Cipher cipher = Cipher.getInstance(“DES/ECB/PKCS5Padding”);
        Cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] encryptedBytes = cipher.doFinal(originalText.getBytes());
        String encryptedText =Base64.getEncoder().encodeToString(encryptedBytes);
        System.out.println(“Original Text: “ + originalText);
        System.out.println(“Encrypted Text: “ + encryptedText);
    }
}
```

Blowfish

```
 import javax.crypto.Cipher;
 import javax.crypto.spec.SecretKeySpec; import
 java.util.Scanner;

 public class Blowfish {
     public static byte[] encrypt(byte[] plaintext, byte[] key) throws Exception { Cipher cipher =
         Cipher.getInstance("Blowfish"); cipher.init(Cipher.ENCRYPT_MODE, new
         SecretKeySpec(key, "Blowfish")); return cipher.doFinal(plaintext);
     }

     public static byte[] decrypt(byte[] ciphertext, byte[] key) throws Exception { Cipher cipher =
         Cipher.getInstance("Blowfish"); cipher.init(Cipher.DECRYPT_MODE, new
         SecretKeySpec(key, "Blowfish")); return cipher.doFinal(ciphertext);
     }
     public static void main(String[] args) throws Exception { byte[] key = "my_secret_key".getBytes();
         Scanner sc = new Scanner(System.in);

         System.out.println("Enter   the   plain   text:   ");
         byte[]    plaintext    =    sc.nextLine().getBytes();
         byte[] ciphertext = encrypt(plaintext, key);
         System.out.println("Ciphertext: " + new String(ciphertext)); byte[]
         decrypted = decrypt(ciphertext, key); System.out.println("Decrypted: " +
         new String(decrypted));
     }
 }
```

Diffie-Hellman

```
import java.util.Scanner;
import java.lang.Math;
public class DiffieHellman{
        public static void main(String args[]){
                Scanner sc = new Scanner(System.in); int P,G,a,b,ka,kb;
                double x,y;
                System.out.println("Enter a Prime: "); P=sc.nextInt();
                System.out.println("Enter a primitive root of " + P +": "); G=sc.nextInt();
                System.out.println("Enter the private key of alice: "); a=sc.nextInt();
                System.out.println("Enter the private key of bob: "); b=sc.nextInt();
                x= (Math.pow(G,a))%P;
                y=(Math.pow(G,b))%P;
                System.out.println("Pub key of alice: "+x);
                System.out.println("Pub key of bob: "+y);
                ka=(int)(Math.pow(y,a))%P;
                kb=(int)(Math.pow(x,b))%P;
                System.out.println("Private shared key: "+ka); System.out.println("Private shared key: "+kb);
         }
}
```

RSA

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
int checkPrime(int n){
	int i;
	int m = n / 2;
	for (i = 2; i <= m; i++) {
		if (n % i == 0){
			return 0; // Not Prime
		}
	}
	return 1; // Prime
}
int findGCD(int n1, int n2){
	int i, gcd;
for(i = 1; i <= n1 && i <= n2; ++i){
	if(n1 % i == 0 && n2 % i == 0)
		gcd = i;
	}
	return gcd;
}
int powMod(int a, int b, int n) {
	long long x = 1, y = a;
	while (b > 0) {
		if (b % 2 == 1)
			x = (x * y) % n;
			y = (y * y) % n; // Squaring the base b /= 2;
	}
	return x % n;
}
int main(int argc, char* argv[]) {
	int p, q;
	int n, phin;
	int data, cipher, decrypt;
	while (1) {
		printf("Enter any two prime numbers: ");
		scanf("%d %d", &p, &q);
		if (!(checkPrime(p) && checkPrime(q)))
			printf("Both numbers are not prime. Please enter prime numbers only...\n");
		else if (!checkPrime(p))
			printf("The first prime number you entered is not prime, please try again…\n");
		else if (!checkPrime(q))
			printf("The second prime number you entered is not prime, please try again...\n");
		else
			break;
	}
	n = p * q;
	phin = (p - 1) * (q - 1); int e = 0;
	for (e = 5; e <= 100; e++) {
		if (findGCD(phin, e) == 1)
			break;
	}
	int d = 0;
	for (d = e + 1; d <= 100; d++) {
		if ( ((d * e) % phin) == 1)
		break;
	}
```

```
        printf("Value of e: %d\nValue of d: %d\n", e, d); printf("Enter some numerical data: "); scanf("%d", &data);
        cipher = powMod(data, e, n);
        printf("The cipher text is: %d\n", cipher); decrypt = powMod(cipher, d, n);
        printf("The decrypted text is: %d\n", decrypt); return 0;
}
```

Or

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
void main(){
        int p=3,q=11,m=5,n,phi,e=7,d,c,M;
        n=p*q;
        phi=(p-1)*(q-1);
        d=1;
        while(((e)*(d))%phi!=1){
                (d)++;
        }
        c=(int)(pow(m,e))%n;
        M=(int)(pow(c,d))%n;
        printf("The Given Text: %d\n",m);
        printf("The Encrypted Text: %d\n",e);
        printf("The Decrypted Text: %d\n",M);
}
```

RC4

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
class RC4Encryption{
        public static byte[]rc4(byte[]key,byte[]data)throws Exception{
                SecretKeySpec keySpec=new SecretKeySpec(key,"RC4");
                Cipher cipher=Cipher.getInstance("RC4"); cipher.init(Cipher.ENCRYPT_MODE,keySpec);
                return cipher.doFinal(data);
        }
        public static byte[]rc4Decrypt(byte[]key,byte[]encryptedData)throws Exception{
                SecretKeySpec keySpec=new SecretKeySpec(key,"RC4");
                Cipher cipher=Cipher.getInstance("RC4"); cipher.init (Cipher.DECRYPT_MODE,keySpec);
                return cipher.doFinal(encryptedData);
        }
        public static void main(String[] args){
                try{
                        String key="secretpassword";
                        String data="heythereiamshadirvan";
                        byte[]encryptedData=rc4(key.getBytes(),data.getBytes());
                        System.out.println("Encrypted Data: "+new String(encryptedData));
                        byte[]decryptedData=rc4Decrypt(key.getBytes(),encryptedData);
                        System.out.println("Decrypted Data: "+new String(decryptedData));
                }
                catch(Exception e){
                        e.printStackTrace();
                }
        }
}
```

MD5 Hash

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
public class md5 {
        public static String getMd5(String input){
                try{
                        MessageDigest md = MessageDigest.getInstance("MD5");
                        byte[] messageDigest = md.digest(input.getBytes()); BigInteger no = new
                        BigInteger(1, messageDigest);
                        String hashtext = no.toString(16);
                        while (hashtext.length() < 32) {
                                hashtext = "0" + hashtext;
                        }
                        return hashtext;
                }
                catch (NoSuchAlgorithmException e) {
                        throw new RuntimeException(e);
                }
        }
        public static void main(String args[]) throws NoSuchAlgorithmException{
                Scanner sc = new Scanner(System.in);
                String s;
                s=sc.nextLine();
                System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
        }
}
```

Digital Signature

```
import java.security. KeyPair:
import java.security. KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.Signature:
import java.util.Scanner;
public class DigitalSignature {
        public static void main(String[] args) {
                try {
                        // Generate a key pair
                        KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("DSA");
                        keyPairGenerator.initialize(2048); KeyPair keyPair =
                        keyPairGenerator.generateKeyPair():
                        // Get the private and public keys
                        PrivateKey privateKey = keyPair.getPrivate();
                        PublicKey publicKey = keyPair.getPublic();
                        // Create a Signature object
                        Signature signature Signature.getInstance("SHA256withDSA");
                        ignature.initSign(privateKey);
                        // Get the message to sign from the user
                        Scanner scanner = new Scanner(System.in);
                        System.out.print("Enter the message to sign: ");
                        String message = scanner.nextLine():
                        // Update the signature object with the message
                signature.update(message.getBytes());
                        // Generate the digital signature
                        byte[] digitalSignature = signature.sign():
                        System.out.println("Digital Signature: " + bytesToHex(digitalSignature));
                        // Verify the signature signature.init Verify(publicKey);
                        signature.update(message.getBytes());
                        boolean isVerified = signature.verify(digitalSignature);
                        if (isVerified) {
                                System.out.println("Signature verified: Message is authentic.");
                        }
                        else {
                                System.out.println("Signature verification failed: Message has been
                        tampered with!");
                        }
                        scanner.close();
                }
                catch (Exception e) {
                         e.printStackTrace();
                }
        }
        / Helper method to convert bytes to hexadecimal string /
        private static String bytesToHex(byte[] bytes) {
                StringBuilder hexString = new StringBuilder();
                for (byte b: bytes) {
                        String hex = Integer.toHexString(0xff & b);
                                if (hex.length() == 1){
                                        hexString.append('0');
                        }
                        hexString.append(hex);
                }
                return hexString.toString();
        }
}
```