1. XOR with 0(HELLO WORLD):

```
#include<stdlib.h>

#include<stdio.h>

#include<string.h>

void bin(unsigned n)

{

  unsigned i;

  for(i=1<<7 ; i>0; i=i/2)

   (n&i) ? printf("1"):printf("0");

}

int main()

{

 int i,n;

 char str[20] = "Hello world";

 char xor_zero[20]= " ";

  printf("       Binary(Str[i])\tBinary(Str[i]) XOR 0\t\n");

  for(i=0;i<strlen(str);i++)

  {

    xor_zero[i] = str[i]^'0';

  printf("%c\t",str[i]);

  bin((int)str[i]);

  printf("\t");

  bin((int)xor_zero[i]);

  printf("\n");

  }

}
```

OUTPUT :

```
            Binary(Str[i])      Binary(Str[i]) XOR 0
H            01001000           01111000
e            01100101           01010101
l            01101100           01011100
l            01101100           01011100
o            01101111           01011111
             00100000           00010000
w            01110111           01000111
o            01101111           01011111
r            01110010           01000010
l         01101100              01011100
d         01100100              01010100
```

2. XOR, AND and OR with 127 (HELLO WORLD):

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
void bin(unsigned n)
{
  unsigned i;
  for(i=1<<7 ; i>0; i=i/2)
   (n&i) ? printf("1"):printf("0");
}
int main()
{
int i,n;
  char str[20] = "Hello world";
  char and[20] = " ";
  char xor_127[20] = " ";
  char or_127[20] = " ";
  printf(" Binary(Str[i])\tBinary(Str[i])XOR_127\tBinary(Str[i])AND_127\tBinary(Str[i])OR_127\t\n");
  for(i=0;i<strlen(str);i++)
  {
   xor_127[i]=str[i]^127;
   and[i]=str[i]&127;
  or_127[i] = str[i]|127;
  printf("%c\t",str[i]);
  bin((int)str[i]);
  printf("\t\t");
  bin((int)xor_127[i]);
  printf("\t\t");
  bin((int)and[i]);
  printf("\t\t");
```

```
 bin((int)or_127[i]);
  printf("\n");
}
}
```

OUTPUT :

```
        Binary(Str[i])              Binary(Str[i])XOR_127     Binary(Str[i])AND_127     Binary(Str[i])OR_127

H       01001000                     00110111                 01001000                        01111111
e       01100101                     00011010                 01100101                        01111111
l       01101100                     00010011                 01101100                        01111111
l       01101100                     00010011                 01101100                        01111111
o       01101111                    00010000                  01101111                        01111111
        00100000                    01011111                  00100000                        01111111
w       01110111                     00001000                 01110111                        01111111
o       01101111                     00010000                 01101111                        01111111
r       01110010                     00001101                 01110010                        01111111
l       01101100                     00010011                 01101100                        01111111
d       01100100                     00011011                 01100100                        01111111
```

3. CIPHER ALGORITHMS

a. Caesar cipher :

```
#include<stdio.h>

#include<ctype.h>

int main()

{

  char text[500], ch;

  int key;

  printf("Enter a message to encrypt: ");

  scanf("%s", text);

  printf("Enter the key: ");

  scanf("%d", & key);

  for (int i = 0; text[i] != '\0'; ++i)

  {

    ch = text[i];

    if (isalnum(ch))

        {

        if (islower(ch))

        {

        ch = (ch - 'a' + key) % 26 + 'a';

      }

      if (isupper(ch))

      {

        ch = (ch - 'A' + key) % 26 + 'A';

      }

      if (isdigit(ch))

      {

        ch = (ch - '0' + key) % 10 + '0';

      }

    }
```

```
else
   {
     printf("Invalid Message");
   }
   text[i] = ch;
 }
 printf("Encrypted message: %s", text);
 printf("\n");
 for (int i = 0; text[i] != '\0'; ++i)
 {
   ch = text[i];
   if (isalnum(ch))
       {
       if (islower(ch))
       {
       ch = (ch - 'a' - key) % 26 + 'a';
     }
     if (isupper(ch))
     {
       ch = (ch - 'A' - key) % 26 + 'A';
     }
     if (isdigit(ch))
     {
       ch = (ch - '0' - key) % 10 + '0';
     }
   }
   else
   {
     printf("Invalid Message");
   }
   text[i] = ch;
```

```
  }
  printf("Decrypted message: %s", text);
  return 0;
}
```

OUTPUT :

```
Enter a message to encrypt: Sravs123
Enter the key: 3
Encrypted message: Vudyv456
Decrypted message: Sravs123
```

b. Substitution Cipher :

```
#include<stdlib.h>
#include<string.h>
#include<stdio.h>
int main()
{
   char str[100];
   printf("Enter the Input String:");
   scanf("%s",str);
   char hash[26]="zyxwvutsrqponmlkjihgfedcba";

   char *e=malloc(strlen(str)*sizeof(char));
   for(int i=0;i<strlen(str);i++)
   {
      e[i]=hash[str[i]-'a'];
   }
   printf("Encrypted message : %s\n",e);
   char *de=malloc(strlen(str)*sizeof(char));
   for(int i=0;i<strlen(e);i++)
   {
      de[i]=hash[e[i]-'a'];
   }
   printf("Decrypted message : %s\n",de);
   return 0;
}
```

OUTPUT :

```
Enter the Input String:mybabe
Encrypted message : nbyzyv
Decrypted message : mybabe
```

c. Hill Cipher :

```
#include<stdio.h>

#include<string.h>

int main()

{

  unsigned int a[3][3] = { { 6, 24, 1 }, { 13, 16, 10 }, { 20, 17, 15 } };

  unsigned int b[3][3] = { { 8, 5, 10 }, { 21, 8, 21 }, { 21, 12, 8 } };

  int i, j;

  unsigned int c[20], d[20];

  char msg[20];

  int determinant = 0, t = 0;

  printf("Enter plain text :\n ");

  scanf("%s", msg);

  for (i = 0; i < 3; i++)

   {

     c[i] = msg[i] - 65;

   }

  for (i = 0; i < 3; i++)

   {

      t = 0;

      for (j = 0; j < 3; j++)

       {

         t = t + (a[i][j] * c[j]);

       }

        d[i] = t % 26;

   }

   printf("\nEncrypted Cipher Text :");

   for (i = 0; i < 3; i++)

   printf(" %c", d[i] + 65);

   for (i = 0; i < 3; i++)

    {
      t = 0;
      for (j = 0; j < 3; j++)
       {
         t = t + (b[i][j] * d[j]);
       }
        c[i] = t % 26;
    }
        printf("\nDecrypted Cipher Text :");
        for (i = 0; i < 3; i++)
        printf(" %c", c[i] + 65);
        return 0;
}
```

OUTPUT :

```
Enter plain text :
MAE
Encrypted Cipher Text : Y O O
Decrypted Cipher Text : M A E
```

4. DES ALGORITHM

```
import javax.crypto.*;

import javax.crypto.spec.*;

import java.nio.charset.StandardCharsets;

import java.util.*;


public class DES {

  private static final String ALGORITHM = "DES/ECB/PKCS5Padding";

   private static final byte[] KEY = "Idio-Reo".getBytes(StandardCharsets.UTF_8);


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

    String plaintext = "MyBabe!";

    byte[] ciphertext = encrypt(plaintext);

    String decrypted = decrypt(ciphertext);

    System.out.println("Plaintext: " + plaintext);

    System.out.println("Ciphertext: " + Base64.getEncoder().encodeToString(ciphertext));

    System.out.println("Decrypted: " + decrypted);

  }


  public static byte[] encrypt(String plaintext) throws Exception {

    Cipher cipher = Cipher.getInstance(ALGORITHM);

    SecretKeySpec keySpec = new SecretKeySpec(KEY, "DES");

    cipher.init(Cipher.ENCRYPT_MODE, keySpec);

    return cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));

  }


  public static String decrypt(byte[] ciphertext) throws Exception {

    Cipher cipher = Cipher.getInstance(ALGORITHM);

    SecretKeySpec keySpec = new SecretKeySpec(KEY, "DES");

    cipher.init(Cipher.DECRYPT_MODE, keySpec);

    byte[] decryptedBytes = cipher.doFinal(ciphertext);
```

```
        return new String(decryptedBytes, StandardCharsets.UTF_8);
    }
}
```

OUTPUT :

```
Plaintext: MyBabe!

Ciphertext: 8vmqyHLGboI=

Decrypted: MyBabe!
```

5. AES ALGORITHM

```
import javax.crypto.*;
import javax.crypto.spec.*;
import java.nio.charset.StandardCharsets;
import java.util.*;

public class AES {
    private static final String ALGORITHM = "AES/ECB/PKCS5Padding";
    private static final byte[] KEY = "0123456789abcdef".getBytes(StandardCharsets.UTF_8);

    public static void main(String[] args) throws Exception {
        String plaintext = "Hello, world!";
        byte[] ciphertext = encrypt(plaintext);
        String decrypted = decrypt(ciphertext);
        System.out.println("Plaintext: " + plaintext);
        System.out.println("Ciphertext: " + Base64.getEncoder().encodeToString(ciphertext));
        System.out.println("Decrypted: " + decrypted);
    }

    public static byte[] encrypt(String plaintext) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        SecretKeySpec keySpec = new SecretKeySpec(KEY, "AES");
        cipher.init(Cipher.ENCRYPT_MODE, keySpec);
        return cipher.doFinal(plaintext.getBytes(StandardCharsets.UTF_8));
    }

    public static String decrypt(byte[] ciphertext) throws Exception {
        Cipher cipher = Cipher.getInstance(ALGORITHM);
        SecretKeySpec keySpec = new SecretKeySpec(KEY, "AES");
        cipher.init(Cipher.DECRYPT_MODE, keySpec);
        byte[] decryptedBytes = cipher.doFinal(ciphertext);
        return new String(decryptedBytes, StandardCharsets.UTF_8);
    }
}
```

OUTPUT :

```
Plaintext: Hello, world!

Ciphertext: yg9b62DJ+X1kYiQWH433FA==

Decrypted: Hello, world!
```

6. RSA(RIVEST SHAMIR ADELMAN ALGORITHM)

```
import java.math.*;

import java.util.*;


class RSA {

        public static void main(String args[])

        {

                int p, q, n, z, d = 0, e, i;

                int msg = 12;

                double c;

                BigInteger msgback;

                p = 3;

                q = 11;

                n = p * q;

                z = (p - 1) * (q - 1);

                System.out.println("the value of z = " + z);

                for (e = 2; e < z; e++)

                {

                        if (gcd(e, z) == 1)

                        {

                                break;

                        }

                }

                System.out.println("the value of e = " + e);

                for (i = 0; i <= 9; i++)

                {

                        int x = 1 + (i * z);

                        if (x % e == 0)

                        {

                                d = x / e;

                                break;
```

```
            }
        }
        System.out.println("the value of d = " + d);
        c = (Math.pow(msg, e)) % n;
        System.out.println("Encrypted message is : " + c);
        BigInteger N = BigInteger.valueOf(n);
        BigInteger C = BigDecimal.valueOf(c).toBigInteger();
        msgback = (C.pow(d)).mod(N);
        System.out.println("Decrypted message is : "
                                + msgback);
    }
    static int gcd(int e, int z)
    {
        if (e == 0)
            return z;
        else
            return gcd(z % e, e);
    }
}
```

OUTPUT:

```
the value of z = 20
the value of e = 3
the value of d = 7
Encrypted message is : 12.0
Decrypted message is : 12
```

7. DIFFIE HELLMAN KEY EXCHANGE

```
import java.util.*;
import java.lang.*;
import java.lang.Math;
public class diffie{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);//k2=aliceComputes
System.out.println("Enter two Large Prime Numbers : ");
int g=sc.nextInt(),n=sc.nextInt();//k1=bobComputes
System.out.println("Enter random numbers X and Y : ");
double x=sc.nextInt(),y=sc.nextInt();//b=bobSends
double A=(Math.pow(g, x)%n),B=(Math.pow(g,y)%n);//a=aliceSends
System.out.println("A value -> "+A+"\nB value -> "+B);
double k1=(Math.pow(B,x)%n),k2=(Math.pow(A,y)%n);
double s = (Math.pow(g,(x*y)))%n;//s=sharedKey
System.out.println("\nk1 value -> "+k1+"\nk2 value -> "+k2);
if ((k2 == s) && (k2 == k1))
System.out.println("Success: Shared Secrets Matches!\nThe secret key : " + s);
else
System.out.println("Error: Shared Secrets does not Match");


}
}
```

OUTPUT:

```
Enter two Large Prime Numbers :
43 79
Enter random numbers X and Y :
2 3
A value -> 32.0
B value -> 33.0

k1 value -> 62.0
k2 value -> 62.0
Success: Shared Secrets Matches!
The secret key : 62.0
```

8. SHA (SECURED HASH ALGORITHM)

```
import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class SHA {

   public static String getSHA(String input)

   {

      try {

         MessageDigest md = MessageDigest.getInstance("SHA-1");

         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

   {

      String s = "Shut up";

      System.out.println("Your HashCode Generated by SHA is: " + getSHA(s));

   }

}
```

OUTPUT :

```
Your HashCode Generated by SHA is: 1505e5c2dda2913b04f46de990f6b7543bd35c2b
```

9. MD5 (MESSAGE DIGEST 5 ALGORITHM)

```
import java.math.BigInteger;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

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

  {

    String s = "GeeksForGeeks";

    System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));

  }

}
```

OUTPUT :

```
Your HashCode Generated by MD5 is: d99637109d197d840fe127b20fa322ff
```

10. DSS (DIGITAL STANDARD SIGNATURE)

```
import java.security.*;
import java.security.spec.*;
import javax.crypto.*;
import java.util.Base64;
public class dss{
  public static void main(String[] args) throws Exception {
    KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
    SecureRandom random = SecureRandom.getInstance("SHA1PRNG");
    keyGen.initialize(1024, random);
    KeyPair pair = keyGen.generateKeyPair();
    PrivateKey privateKey = pair.getPrivate();
    PublicKey publicKey = pair.getPublic();
    Signature dsa = Signature.getInstance("SHA256withDSA");
    String message = "Shut up";
    dsa.initSign(privateKey);
    dsa.update(message.getBytes());
    byte[] signature = dsa.sign();
    dsa.initVerify(publicKey);
    dsa.update(message.getBytes());
    boolean verified = dsa.verify(signature);
    System.out.println("Message: " + message);
    System.out.println("Signature: " + Base64.getEncoder().encodeToString(signature));
    System.out.println("Signature verified: " + verified);
  }
}
```

OUTPUT :

```
Message: Shut up
Signature: MC0CFQCBe7YI9a03ZAkbeiadB+7vpwkkFwIUBFg0EarMfZRXnn6Tj858EpfFHQ4=
Signature verified: true
```