1.

```
#include <stdio.h>

void main() {
  char text[] = "Hello world";
  printf("Input: %s \n", text);

  for (char *ptr = text; *ptr != '\0'; ptr++)
    *ptr = *ptr ^ 0;
  // Ptr is a pointer that will give us each char
  // of 'text' one by one
  // \0 represents end of string in C, thus we
  // are using it in our condition
  // ptr++ will make the pointer point to the next char
  // in the text string

  printf("After XOR: %s \n", text);
}
```

2.

```
#include <stdio.h>
#include <string.h>

void main() {
  char text[] = "Hello world";
  char *text1 = strdup(text);
  // Equivalent to char text1[] = "Hello world"
  char *text2 = strdup(text);

  printf("Input: %s \n", text);

  // Three pointers for three strings
  char *p1 = text, *p2 = text1, *p3 = text2;

  // Since the three strings are of equal length
  // we can just use p1 in the loop condition
  while (*p1 != '\0') {
    *p1 = *p1 ^ 127;
    *p2 = *p2 & 127;
    *p3 = *p3 | 127;
    p1++, p2++, p3++;
  }

  printf("XOR output: %s \n", text);
  printf("AND output: %s \n", text1);
  printf("OR output: %s \n", text2);
}
```

3.

i. subscrypt:

```
// This program handles both Caesar and Substitution Ciphers.
class SubsCrypt {
    public static void main(String args[]) {
            String text = "A good press is the backbone for a true Democracy.";
            System.out.println("Text:\n"+ text);
            String cipherText = SubsCrypt.caesarEncrypt(text);
            System.out.println("CipherText (Caesar):\n"+ cipherText);
```

```
        System.out.println("Decryption:\n"+ SubsCrypt.caesarDecrypt(cipherText));

        int key = 23;
        cipherText = SubsCrypt.substitutionEncrypt(text, key);
        System.out.printf("CipherText(Substitution) with key %d: \n%s \n", key, cipherText);
        System.out.println("Decryption:\n"+ SubsCrypt.substitutionDecrypt(cipherText, key));
        System.out.println();
    }

    static String caesarEncrypt(String message) {
        return operate(message, 3, true);
    };
    static String substitutionEncrypt(String message, int key) {
        return operate(message, key, true);
    }
    static String caesarDecrypt(String message) {
        return operate(message, 3, false);
    };
    static String substitutionDecrypt(String message, int key) {
        return operate(message, key, false);
    }
    static String operate(String message, int key, boolean encrypt) {
        String cipher = new String();
        for (int i=0; i<message.length(); i++) {
            int c = message.charAt(i);
            if (Character.isAlphabetic(c) == false) {
                cipher += (char)c;
                continue;
            }
            c = Character.toLowerCase(c) - 'a';

            if (encrypt == true)
                c = c + key;
            else
                c = c - key;

            if (c < 0)
                c = 26 + c;
            else
                c = c % 26;

            cipher += (char)(c + 'a');
        }
        return cipher;
    }
}
```

ii. playfair

```
def prepare_text(text):
    # Remove spaces and convert to uppercase
    text = text.replace(" ", "").upper()
    # Replace 'J' with 'I'
    text = text.replace("J", "I")
    return text

def generate_key_matrix(key):
    key = prepare_text(key)
    alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ"
```

```
    key_matrix = []

    for char in key:
        if char not in key_matrix:
            key_matrix.append(char)

    for char in alphabet:
        if char not in key_matrix:
            key_matrix.append(char)

    key_matrix = [key_matrix[i:i+5] for i in range(0, 25, 5)]
    return key_matrix

def find_position(matrix, char):
    for i in range(5):
        for j in range(5):
            if matrix[i][j] == char:
                return i, j

def encrypt(plaintext, key):
    plaintext = prepare_text(plaintext)
    key_matrix = generate_key_matrix(key)
    ciphertext = ""

    for i in range(0, len(plaintext), 2):
        char1 = plaintext[i]
        char2 = plaintext[i + 1] if i + 1 < len(plaintext) else 'X'

        row1, col1 = find_position(key_matrix, char1)
        row2, col2 = find_position(key_matrix, char2)

        if row1 == row2:
            ciphertext += key_matrix[row1][(col1 + 1) % 5] + key_matrix[row2][(col2 + 1) % 5]
        elif col1 == col2:
            ciphertext += key_matrix[(row1 + 1) % 5][col1] + key_matrix[(row2 + 1) % 5][col2]
        else:
            ciphertext += key_matrix[row1][col2] + key_matrix[row2][col1]

    return ciphertext

def decrypt(ciphertext, key):
    ciphertext = prepare_text(ciphertext)
    key_matrix = generate_key_matrix(key)
    plaintext = ""

    for i in range(0, len(ciphertext), 2):
        char1 = ciphertext[i]
        char2 = ciphertext[i + 1] if i + 1 < len(ciphertext) else 'X'

        row1, col1 = find_position(key_matrix, char1)
        row2, col2 = find_position(key_matrix, char2)

        if row1 == row2:
            plaintext += key_matrix[row1][(col1 - 1) % 5] + key_matrix[row2][(col2 - 1) % 5]
        elif col1 == col2:
            plaintext += key_matrix[(row1 - 1) % 5][col1] + key_matrix[(row2 - 1) % 5][col2]
        else:
            plaintext += key_matrix[row1][col2] + key_matrix[row2][col1]
```

```
    return plaintext

# Example usage:
key = "KEYWORD"
plaintext = "HELLO EARTH"

encrypted_text = encrypt(plaintext, key)
print("Encrypted:", encrypted_text)

decrypted_text = decrypt(encrypted_text, key)
print("Decrypted:", decrypted_text)
```

iii. <u>hill:</u>

```
import numpy as np

def prepare_text(text, block_size):
    # Remove spaces and convert to uppercase
    text = text.replace(" ", "").upper()
    # Pad the text with 'X' if needed
    if len(text) % block_size != 0:
        text += 'X' * (block_size - len(text) % block_size)
    return text

def text_to_matrix(text, block_size):
    matrix = []
    for i in range(0, len(text), block_size):
        block = [ord(char) - ord('A') for char in text[i:i+block_size]]
        matrix.append(block)
    return matrix

def matrix_to_text(matrix):
    text = ""
    for row in matrix:
        text += ''.join([chr(char + ord('A')) for char in row])
    return text

def matrix_inverse(matrix, mod):
    # Calculate the inverse of a matrix in modulo mod
    det = int(np.linalg.det(matrix))
    adjugate = np.round(np.linalg.inv(matrix) * det) % mod
    inverse = (det * np.round(np.linalg.inv(matrix) * det) % mod).astype(int)
    return inverse

def encrypt(plaintext, key_matrix, mod):
    plaintext = prepare_text(plaintext, len(key_matrix))
    plaintext_matrix = text_to_matrix(plaintext, len(key_matrix))

    result_matrix = np.dot(plaintext_matrix, key_matrix) % mod
    ciphertext = matrix_to_text(result_matrix)

    return ciphertext

def decrypt(ciphertext, key_matrix, mod):
    ciphertext = prepare_text(ciphertext, len(key_matrix))
    ciphertext_matrix = text_to_matrix(ciphertext, len(key_matrix))
```

```
    key_matrix_inv = matrix_inverse(key_matrix, mod)
    result_matrix = np.dot(ciphertext_matrix, key_matrix_inv) % mod
    plaintext = matrix_to_text(result_matrix)

    return plaintext

# Example usage:
key_matrix = np.array([[6, 24, 1], [13, 16, 10], [20, 17, 15]])
mod = 26  # The modulo value for the alphabet size

plaintext = "RAJENDA"
encrypted_text = encrypt(plaintext, key_matrix, mod)
print("Encrypted:", encrypted_text)

decrypted_text = decrypt(encrypted_text, key_matrix, mod)
print("Decrypted:", decrypted_text)
```

6: AES

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class AES_Builtin {

    public static void main(String[] args)
            throws Exception
    {
            String message = "This is my input string";
            System.out.println("Input: " + message);

            //Get the keyGenerator
            KeyGenerator kgen = KeyGenerator.getInstance("AES");
            kgen.init(128);

            //Generate secret key specs
            SecretKey skey = kgen.generateKey();
            byte[] raw = skey.getEncoded();
            SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");

            //Instantiate the cipher
            Cipher cipher = Cipher.getInstance("AES");
            cipher.init(Cipher.ENCRYPT_MODE, skeySpec);

            // Encryption
            byte[] encrypted = cipher.doFinal(message.getBytes());
            String cipherString = new String(encrypted);
            System.out.println("Encrypted String: " + cipherString);

            // Decryption
            cipher.init(Cipher.DECRYPT_MODE, skeySpec);
            byte[] original = cipher.doFinal(encrypted);
            String originalString = new String(original);
            System.out.println("Decrypted: " + originalString);
    }
}
```

7. RC4:

```
// RC4 algorithm using Java cryptography
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;

public class RC4 {

   public static void main(String[] args)
         throws Exception
   {
         String message = "This is my input string";
         String key = "I am a key";

         System.out.println("Input: " + message);
         System.out.println("Key: " + key);
         SecretKeySpec keySpec = new SecretKeySpec(key.getBytes(), "RC4");

         // Instantiate the cipher
         Cipher cipher = Cipher.getInstance("RC4");
         cipher.init(Cipher.ENCRYPT_MODE, keySpec);

         // Encryption
         byte[] cipherText = cipher.doFinal(message.getBytes());
         String cipherString = new String(cipherText);
         System.out.println("Encrypted String: " + cipherString);

         // Decryption
         cipher.init(Cipher.DECRYPT_MODE, keySpec);
         byte[] original = cipher.doFinal(cipherText);
         String originalString = new String(original);
         System.out.println("Decrypted: " + originalString);
   }
}
```

8. RSA:

```
import java.math.BigInteger;
class RSA
{
   public static void main(String args[])
   {
         int msg = 420;
         int p = 17;
         int q = 31;
         int n = p * q;
         int phi = (p - 1) * (q - 1);

         int e;
         for (e = 2; e < phi; e++)
            if (gcd(e, phi) == 1)
                  break;

         int d = 1;
         for (int i=0; i<10; i++) {
            int x = i*phi + 1;
            if (x % e == 0) {
                  d = x / e;
```

```
                break;
            }
        }

        long c = (long)Math.pow(msg, e) % n;

        // We need BigInteger for very large numbers here
        BigInteger C = BigInteger.valueOf(c);
        BigInteger N = BigInteger.valueOf(n);
        BigInteger D = BigInteger.valueOf(d);
        BigInteger decrypted = C.modPow(D, N);

        System.out.println("Value of P: " + p);
        System.out.println("Value of Q: " + q);
        System.out.println("Value of Phi: " + phi);
        System.out.println("Value of E: " + e);
        System.out.println("Value of D: " + d);

        System.out.println("Input Message: " + msg);
        System.out.println("Encrypted Message: " + c);
        System.out.println("Decrypted message: " + decrypted);
    }

    static int gcd(int e, int z)
    {
        if (e == 0)
            return z;
        else
            return gcd(z%e, e);
    }
}
```

9. Diffie Hellman

```
<!DOCTYPE html>
<html>
  <head>
    <title> Diffie Hellman Page </title>
    <style>

    </style>
    <script>

      function diffie_hellman() {
        let p = document.getElementById("pVal").value;
        let g = document.getElementById("gVal").value;

        let a = document.getElementById("pka").value;
        let b = document.getElementById("pkb").value;

        let x = Math.pow(g, a) % p;
        let y = Math.pow(g, b) % p;

        let ka = Math.pow(y, a) % p;
        let kb = Math.pow(x, b) % p;

        document.getElementById("output1").innerHTML = ka
        document.getElementById("output2").innerHTML = kb
      }
```

```
  </script>
</head>
<body>
  <div class="container">
    <h2 style="text-align: center;">Diffie Hellman</h2>
      <label for="pVal">Value of P:</label>
      <input type="text" id="pVal" required>

      <label for="gVal">Value of G (Primitive root of P):</label>
      <input type="text" id="gVal" required>

      <label for="pka">Private key a for User1:</label>
      <input type="text" id="pka" required>

      <label for="pkb">Private key b for User2:</label>
      <input type="text" id="pkb" required>

      <button id="submitButton" onclick="diffie_hellman()">Submit</button>

      <p>Secret key for User1: <span id="output1"></p></p>
      <p>Secret key for User2: <span id="output2"></p></p>
  </div>

</body>
</html>
```

10. SHA1

```
// Java program to calculate SHA-1 hash value

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA1 {
        public static String encryptThisString(String input)
        {
                try {
                        MessageDigest md = MessageDigest.getInstance("SHA-1");
                        byte[] messageDigest = md.digest(input.getBytes());
                        BigInteger no = new BigInteger(1, messageDigest);

                        // Convert message digest into hex value
                        String hashtext = no.toString(16);

                        // Add preceding 0s to make it 32 bit
                        while (hashtext.length() < 32) {
                                hashtext = "0" + hashtext;
                        }

                        // return the HashText
                        return hashtext;
                }

                // For specifying wrong message digest algorithms
                catch (NoSuchAlgorithmException e) {
                        throw new RuntimeException(e);
                }
```

```
        }

        // Driver code
        public static void main(String args[]) throws NoSuchAlgorithmException
        {

                System.out.println("HashCode Generated by SHA-1 for: ");

                String s1 = "Rajendar";
                System.out.println("\n" + s1 + " : " + encryptThisString(s1));

                String s2 = "hello world";
                System.out.println("\n" + s2 + " : " + encryptThisString(s2));
        }
}
```

11. MD5:

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

// Java program to calculate MD5 hash value
public class MD5 {
        public static String getMd5(String input)
        {
                try {

                        // Static getInstance method is called with hashing MD5
                        MessageDigest md = MessageDigest.getInstance("MD5");

                        // digest() method is called to calculate message digest
                        // of an input digest() return array of byte
                        byte[] messageDigest = md.digest(input.getBytes());

                        // Convert byte array into signum representation
                        BigInteger no = new BigInteger(1, messageDigest);

                        // Convert message digest into hex value
                        String hashtext = no.toString(16);
                        while (hashtext.length() < 32) {
                                hashtext = "0" + hashtext;
                        }
                        return hashtext;
                }

                // For specifying wrong message digest algorithms
                catch (NoSuchAlgorithmException e) {
                        throw new RuntimeException(e);
                }
        }
        // Driver code
        public static void main(String args[]) throws NoSuchAlgorithmException
        {
                String s = "Enugula Rajendar";
                System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
        }
}
```

12 columnar:

```
import java.util.Arrays;

public class Columnar {

   public static void main(String[] args) {
      String msg = "Hello, CSE5!";
      String key = "HACK";

      String cipher = encrypt(msg, key);
      System.out.println("PlainText: " + msg);
      System.out.println("Key: " + key);
      System.out.println("Encrypted Message: " + cipher);

      String decryptedMessage = decrypt(cipher, key);
      System.out.println("Decrypted Message: " + decryptedMessage);
   }

   public static
         String encrypt(String msg, String key)
   {
      int msgLen = msg.length();
      char[] msgArr = msg.toCharArray();

      int cols = key.length();
      int rows = msgLen / cols;
          // If there is any remaining space, add an extra row
          if (msgLen % cols != 0)
             rows++;

      char[][] keyMatrix = new char[rows][cols];
          int i, j, k=0;

          // Create the keyMatrix by copying the message followed
          // by padding '-'
          for (i=0; i < rows; i++)
             for (j=0; j < cols; j++) {
                    if (k < msgLen) {
                       keyMatrix[i][j] = msgArr[k];
                       k++;
                    } else
                       keyMatrix[i][j] = '-';
             }

      StringBuilder cipher = new StringBuilder();
      char[] sortedKey = key.toCharArray(); // Alphabetically sorted
      Arrays.sort(sortedKey);

      for (i = 0; i < cols; i++) {
             // get the column index in alphabetical order of key string
         int index = key.indexOf(sortedKey[i]);

             // Fill the cipher string with the elements in that obtained column
         for (char[] row : keyMatrix) {
            cipher.append(row[index]);
         }
      }
```

```
        return cipher.toString();
    }

    public static String
            decrypt(String cipher, String key)
    {
        int msgLen = cipher.length();
        char[] cipherArr = cipher.toCharArray();

        int cols = key.length();
        int rows = msgLen / cols;
            if (msgLen % cols != 0)
                rows++;

        char[][] keyMatrix = new char[rows][cols];
        char[] sortedKey = key.toCharArray();
        Arrays.sort(sortedKey);
        int msgIndex = 0;

        for (int i = 0; i < cols; i++) {
            int index = key.indexOf(sortedKey[i]);

            for (int j = 0; j < rows; j++) {
                keyMatrix[j][index] = cipherArr[msgIndex];
                msgIndex++;
            }
        }

        StringBuilder msg = new StringBuilder();

        for (char[] row : keyMatrix)
            for (char c : row)
                        msg.append(c);

        return msg.toString();
    }
}
```

adv columnar:

```
/*
  Advanced Columnar Cipher
  This cipher performs columnar cipher n number of times
  This program depends on Columnar.java to be implemented
 */
class Adv_Columnar {
    public static void main(String[] args) {
        String message = "Hello CSE5";
        String key = "megabuck";
        int iterations = 9;

        Adv_Columnar advC = new Adv_Columnar();

        String cipher = advC.encrypt(message, key, iterations);
        String decrypted = advC.decrypt(cipher, key, iterations);

        System.out.println("Message: " + message);
        System.out.println("Key: " + key);
        System.out.println("Iterations: " + iterations);
```

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

    }

    public String
            encrypt(String message,
                    String key, int iterations)
    {
        String cipher = message;

            // Note Columnar.java is required for this
        for (int i=0; i<iterations; i++)
            cipher = Columnar.encrypt(cipher, key);

        return cipher;
    }

    public String
            decrypt(String cipher, String key,
                    int iterations)
    {
        String message = cipher;

        for (int i=0; i<iterations; i++)
            message = Columnar.decrypt(message, key);

            message = message.replace('-', ' ');

        return message;
    }


}
```

12. <u>euclidean:</u>

```
// Java program to demonstrate working of extended
// Euclidean Algorithm

class Euclid {
    // Euclidean Algorithm for GCD

    static int
            gcd(int a, int b)
    {
            if (a == 0)
                return b;
            else
                return gcd(b%a, a);
    }

    // Driver Program
    public static void main(String[] args)
    {
            int x = 1, y = 1;
            int a = 35, b = 15;
            int g = gcd(a, b);
```

```
        System.out.println("Given inputs: "+ a +", "+ b);
        System.out.println("gcd(" + a + " , " + b
                           + ") = " + g);
    }
}
```

ext euclidean:

```
// Java program to demonstrate working of extended
// Euclidean Algorithm

class Euclid_Extended {
    static public void gcdExtended(long a, long b)
    {
        long x = 0, y = 1, lastx = 1, lasty = 0, temp;
        while (b != 0) {
            long q = a / b;
            long r = a % b;

            a = b;
            b = r;

            temp = x;
            x = lastx - q * x;
            lastx = temp;

            temp = y;
            y = lasty - q * y;
            lasty = temp;
        }
        System.out.println("GCD "+ a +" and its Roots x : "+ lastx +" y :"+ lasty);
        System.out.println("Final Equation:");
        System.out.printf("%dx + %dy = %d \n", lastx, lasty, a);
    }

    // Driver Program
    public static void main(String[] args)
    {
        long a = 35, b = 15;
        //this will print result like
        //Roots x : 1 y :-2
        System.out.println("Given a and b for ax+by = GCD(a,b) are:");
        System.out.printf("a = %d \t b = %d \n", a, b);
        gcdExtended(a, b);

    }
}
```