Ceaser Cipher:-

```
def encrypt(text,key):
    result = ""
    for i in range(len(text)):
        if(text(i).isupper()):
            result += chr((ord(text(i)) + key-65)%26+65)
        else:
            result += chr((ord(text(i)) + key-97)%26+97)
    return result

if __name__=="__main__":
    print(encrypt(input("Enter Plaintext:"),int(input("Enter key:"))))
```

Atbash Cipher:-

```
lookup_table = {'A' : 'Z', 'B' : 'Y', 'C' : 'X', 'D' : 'W', 'E' : 'V',
        'F' : 'U', 'G' : 'T', 'H' : 'S', 'I' : 'R', 'J' : 'Q',
        'K' : 'P', 'L' : 'O', 'M' : 'N', 'N' : 'M', 'O' : 'L',
        'P' : 'K', 'Q' : 'J', 'R' : 'I', 'S' : 'H', 'T' : 'G',
        'U' : 'F', 'V' : 'E', 'W' : 'D', 'X' : 'C', 'Y' : 'B', 'Z' : 'A'}

def encrypt(message):
    cipher = ""
    for i in message:
        if(i != " "):
            cipher = cipher + lookup_table(i)
        else:
            cipher += " "
    return cipher

if __name__=="__main__":
    msg = input("Enter message:")
    print(encrypt(msg.upper()))
```

Product Cipher:-

```
lookup_table = {'A' : 'Z', 'B' : 'Y', 'C' : 'X', 'D' : 'W', 'E' : 'V',
    'F' : 'U', 'G' : 'T', 'H' : 'S', 'I' : 'R', 'J' : 'Q',
    'K' : 'P', 'L' : 'O', 'M' : 'N', 'N' : 'M', 'O' : 'L',
    'P' : 'K', 'Q' : 'J', 'R' : 'I', 'S' : 'H', 'T' : 'G',
    'U' : 'F', 'V' : 'E', 'W' : 'D', 'X' : 'C', 'Y' : 'B', 'Z' : 'A'}


def encryptAtbash(message):
  cipher = ""
  for i in message:
    if(i != " "):
      cipher = cipher + lookup_table(i)
    else:
      cipher += " "
  return cipher


def encryptCeaser(text,key):
  result = ""
  for i in range(len(text)):
    result += chr((ord(text(i)) + key-65)%26+65)
  return result


if __name__=="__main__":
  print("———Atbash Cipher———")
  msg = input("Enter message:")
  atbash = encryptAtbash(msg.upper())
  print(f"Atbash Ciphered Text:{atbash}")
  print("———Ceaser Cipher———")
  ceaser = encryptCeaser(atbash,int(input("Enter the key:")))
  print(f"Ceaser Ciphered Text:{ceaser}")
```

Railfence Cipher:-

```
def encrypt(text,key):
    rail = (('\n' for i in range(len(text)))
                for i in range(key))
    dir_down = False
    row,column = 0,0
    for i in range(len(text)):
        if(row==0)or(row==key-1):
            dir_down = not dir_down
        rail(row)(column) = text(i)
        column += 1
        if(dir_down):
            row += 1
        else:
            row -= 1
    result = ()
    for i in range(key):
        for j in range(len(text)):
            if(rail(i)(j) != "\n"):
                result.append(rail(i)(j))
    return ("".join(result))

if __name__=="__main__":
    print(encrypt(input("Enter Plaintext:"),int(input("Enter key:"))))
```

RSA Algorithm:-

```
import math
print("Step 1: Enter you prime numbers:")
p = int(input(">>"))
q = int(input(">>"))
print("Step 2: Calculate value of n and phi:")
n = p*q
phi = (p-1)*(q-1)
print(f"n = {n}")
print(f"phi = {phi}")
print("Step 3: Calculate value of e such that 1<e<phi:")
e=2
while(e<phi):
    if(math.gcd(e,phi)==1):
        break;
    else:
        e += 1
print(f"e = {e}")
print("Step 4: Calculate value of d:")
d = (1 + (2*phi))/e
print(f"d = {d}")
print("Step 5: Encryption and Decryption of Plaintext:")
msg = int(input("Enter a number:"))
c = math.fmod(pow(e,msg),n)
print("Encrypted Text:",c)
print("Original Message:",math.fmod(pow(d,msg),n))
```

MD5 Message Digest:-

```
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
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 = "Sarthak Rane";
    System.out.println("Your HashCode Generated by MD5 is: " + getMd5(s));
  }
}
```