Ex 1

```
import math

def encryptMessage(key, message):

   cipherText = [""] * key

   for col in range(key):

      pointer = col

      while pointer < len(message):

         cipherText[col] += message[pointer]

         pointer += key

   return "".join(cipherText)


def decryptMessage(key, message):

   numCols = math.ceil(len(message)/key)

   numRows = key

   numShadedBoxes = (numCols * numRows) - len(message)

   plainText = [""] * numCols

   col = 0

   row = 0

   for symbol in message:

      plainText[col] += symbol

      col += 1

      if ((col == numCols) or (col == numCols-1) and (row >= numRows-numShadedBoxes)):

         col = 0

         row += 1

   return "".join(plainText)
```

```
message = input("Enter message: ")

key = int(input("Enter key [2-%s]: " % (len(message) - 1)))

mode = input("Encryption/Decryption [e/d]: ")

if mode.lower().startswith("e"):

    text = encryptMessage(key, message)

elif mode.lower().startswith("d"):

    text = decryptMessage(key, message)

print("Output:",text)
```

ex2

```
import math


def encryptMessage(key, message):

    cipherText = [""] * key

    for col in range(key):

        pointer = col

        while pointer < len(message):

            cipherText[col] += message[pointer]

            pointer += key

    return "".join(cipherText)


def decryptMessage(key, message):

    numCols = math.ceil(len(message)/key)

    numRows = key
    numShadedBoxes = (numCols * numRows) - len(message)
    plainText = [""] * numCols
    col = 0
    row = 0
    for symbol in message:
        plainText[col] += symbol
        col += 1
        if ((col == numCols) or (col == numCols-1) and (row >= numRows-numShadedBoxes)):
            col = 0
            row += 1
    return "".join(plainText)

message = input("Enter message: ")
key = int(input("Enter key [2-%s]: " % (len(message) - 1)))
mode = input("Encryption/Decryption [e/d]: ")
if mode.lower().startswith("e"):
    text = encryptMessage(key, message)
elif mode.lower().startswith("d"):
    text = decryptMessage(key, message)
print("Output:",text)
```

Ex-3

```
from Crypto.PublicKey import DSA
from Crypto.Hash import SHA256

from Crypto.Signature import DSS

KEYSIZE = 1024

msg = input("Enter your message: ")

msg1 = msg+'new'

message = msg.encode()

message1 =  msg1.encode()

key = DSA.generate(KEYSIZE)

publickey = key.publickey()

print(publickey.exportKey())

message_hash = SHA256.new(message)

message_hash1 = SHA256.new(message1)

signer = DSS.new(key, 'fips-186-3')

signature = signer.sign(message_hash)

print(int.from_bytes(signature, "big", signed=False))

verifier = DSS.new(publickey, 'fips-186-3')

try:

 verifier.verify(message_hash, signature)

 print("Verification successful")

except:

        print("Verification failed")
```

Ex-4

```
def caesar_encryption(plaintext,key):
  encryption_str = ''
  for i in plaintext:
    if i.isupper():
      temp = 65 + ((ord(i) - 65 + key) % 26)
      encryption_str = encryption_str + chr(temp)
    elif i.islower():
      temp = 97 + ((ord(i) - 97 + key) % 26)
      encryption_str = encryption_str + chr(temp)
    else:
      encryption_str = encryption_str + i
    print("The ciphertext is:",encryption_str)
plaintext = input("Enter the encryption message:")
caesar_encryption(plaintext,13)
def caesar_decryption(plaintext,key):
  decryption_str = ''
  for i in plaintext:
    if i.isupper():
      temp = 65 + ((ord(i) - 65 - key) % 26)
      decryption_str = decryption_str + chr(temp)
    elif i.islower():
      temp = 97 + ((ord(i) - 97 - key) % 26)
      decryption_str = decryption_str + chr(temp)
    else:
      decryption_str = decryption_str + i
    print("The ciphertext is:",decryption_str)
plaintext = input("Enter the decrption message:")
caesar_decryption(plaintext,13)
```