MCQ

1. a - Symmetoric key encryption with receiver public key.
2. c - Spyware.
3. c - An authentication of on electronic record.
4. d - none.
5. a - Only on alphanumeric.
6. c - All.
7. a - Hash value.
8. d - option a and c are right.
9. b - to make even no. of letters.
10. c - Possibility of replacement.

Q3 Vigenere Cipher -

input plain text = "Cryptography"
key = "Monarchy"

```
def key2 (string, key):
    key = list (key)
    If len (string) == len (key):
        return (key)
    else:
        for i in range (len (string) - len(key)):
            key.append (key[i % len (key)])
    return (" ".join (key))

def enc (string, key):
    enresult = [ ]
    for i in range (len (string)):
        x = (ord (string [i]) + ord (key [i])) % 26
        x = x + ord ('A')
        enresult.append (chr (x))
    return (" ".join ( enresult ))
```

```
def dec(enresult, key):
    deresult = []
    for i in range (len ( enresult )):
        x = (ord (enresult [i]) - ord (key [i]) +26)
                                               % 26

        x = x + ord ('A')
        deresult . append (chr (x))
    return (" ". join (deresult))

String = "CRYPTOGRAPHY"
key1 = "MONARCHY"
key = key2 (String, key1)
enresult = enc (String , key)
print ("Cipher Text after encrypting =", enresult)
print ("Plain Text after decrypting =", dec(enresult, key))
```

Q4 WAP to implement OTP-

```
import math, random
def gen ( ):
    no = "0123456789"
    otp = " "
    for i in range (8):
        otp = otp + no[math.floor (random.random()
                                        * 10)]

    return otp
print ("Your 8 digit OTP is =", gen ())
```

Q5 Caeser Cipher -

input Plain Text = "ATTACK FROM NORTH"
key - 3

```
def enc (string):
  enresult = "  "
  for char in string:
      if char == " ":
          enresult = enresult + char
      elif char.isupper ():
          enresult = enresult + char ((ord(char) + 3
                             -65) % 26 + 65)
      else:
          enresult = enresult + char ((ord(char) +3
                             -97) % +97)
      return enresult

def dec (string):
  deresult = "  "
  for char in string:
    if char == "  ":
       deresult = deresult + char
    elif char . is upper ():
       deresult = deresult + char (ord(char) - 3 - 65)
                             % 26 +65)
```

```
else :
    deresult = deresult + char ((ord (char) - 3 - 97) %
                                        26 + 97)
    return deresult

text = " ATTACK FROM NORTH"
print (" Cipher Text after encryption =", enc (text))
print (" Plain Text after decryption =", dec (enresult))
```