Name: Akash Bhandari

Roll no: 1121008 (BCA '6th A'

(1)

MCQ Answer

Ans-1 Asymmetric key encryption with Senders Public Key

Ans-2 Spyware

Ans-3 An authentication of an electronic record

Ans-4 Cyber laws

Ans-5 only on alphanumeric

Ans-6 Idea is same title is different

Ans-7 Checksum

Ans-8 The identifier of the character is changed while its position remain unchanged.

Ans-9 both b and c

Ans-10 none

Akash

Ans-3 Vignere

```
def generateKey(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(encrypt_text))

def encryption(string, Key):
    encrypt_text = []
    for i is range(len(string)):
        x = (ord(string[i]) + ord(Key[i])) % 26
        x = x + ord('A')
        encrypt_text.append(chr(x))
    return("", Join(encrypt_text))
```

```
def decryption (encrypt_text, key):
    Original_text = []
    for i in range (len(string)):
        x = (ord (string[i]) + ord (key[i])) %26
        x = x + 65
        Original_text.append (chr(x))
    return ("", join (original_text))

If __name__ == "__main__":
        s = "Cryptography"
        ~~key~~ string = s.upper()
        ~~Key = generateKey~~
            Keyword = "Monarchy"
        Key = generate key (string, keyword)
        encrypt_text = encryption (string, key)
        Print ("Encrypted msg:", encrypt_text)
```

Akash

Akash - Bhandari - 1121008

(4)

Ans-4 OTP

```
import math, random
def generateotp():
    x = "0123456789"
    OTP = ""
for i in range(4):
OTP = OTP + x [math.floor(random.random() = 10)]
    return OTP

if __name__ = "__main__":
Priutf ("OTP of 4 digit is", generate.otP())
```

Akash

Ans-5 Encryption and decryption using caesar cipher

```
def encryption (plain_text, key):
        encrypted = ""
        for c in plaintext:
    if c.isupper():
c_index = ord(c) - ord('A')
 c_shifted = (c_index + key) % 26 + ord('A')
    c_new = chr(c_shifted)
    encrypted += c_new
 def c.islower():
    c_index = ord(c) - ord('a')
    c_shifted = (c_index + key) % 26 + ord('a')
    c_new = chr(c_shifted)
      encrypted += c_new
  else c.isdigit():
      c_new = (int(c) + key) % 10
        encrypted += str(c_new)
    else:
        encrypted += c
return encrypted
def decryption (ciphertext, key):
        decrypted = ""
    for c in ciphertext:
      if c.isupper():
    c_index = ord(c) - ord('A')
    c_og_pos = (c_index - key) % 26 + ord('A')
```

Akash

```
            c.ol = chr ( c.os.pos)
            decrypted += c.os
        elif c.islower ():
                c.index = ord(c)-ord('a')
                c.os.pos = (c.index - key) % 26 + ord ('a')
                c.os = chr ( c.os.pos)
                decrypted += c.os
            elif c.isdigit ():
                c.os = (int (c) - key) % 10
                decrypted = str ( c.os)
            else:
                decrypted += c
            return decrypted

plain_text = "Attack from North"
ciphertext = encryption (plain_text, 4)
print ( "Plain text message: \n", plain_text)
print ("Encrypted ciphertext: \n", ciphertext)

decrypted msg = decryption ( ciphertext, 4)
print ("The decrypted message is: \n", decrypting)
```

Akash