NAME: AKASH COURSE: BCA-A SEM-6 ROLL NO- 1121007
SUB- INFO SECURITY AND CYBER LAWS PRACTICAL

⇒ MCQs

1) Asymmetric key encryption with Sender public key
2) c. Spyware
3) An authentication of an electronic record
4) c. Cyber laws
5) a. Only on alphanumeric
6) Idea in Same ~~title~~ title is different
7) a. hash Value
8) a. The position of the character is changed inspite of its identity.
9) both b and c
10) none

NAME: AKASH Course: BCA-A-6 Roll No: 1121007

Q3 V: Genere Cipher

```
def generate key(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))
cipher text(String, key):
cipher.Text = [ ]
for i in range (len(String)):
    x = (ord String[i] + ord (key[i])) % 26
    x = 0d(
        x + = Ord("A")
```

```
        cipher_text.append(chr(x))
    return("".join(cipher_text))
# function for decrypting
def Original text(cipher_text, key):
    Orig_text = []
for i in range(len(cipher_text)):
    x=(ord(cipher_text[i])- ord(key[i])+26) %26

    x+ = ord('A')
    Orig_text.append(chr(x))
    return("".join(orig_text))
# Driver Code
if __name__ == "__main__":
    String = "Cryptography"
    Keyword = "monarchy"
    key = generate key(String, keyword)
    Print("ciphertext:", cipher_text)
Print("Original Decrypted text:", Original text(
                                    Cipher_text, key))
```

NAME: AKASH COURSE: BCA-6-A ROLL NO- 1121007

Q4

```
# import library
import math.random
# function to Generate OTP
def generate OTP():
# Declare a digits Variable
# which Stores all digits
digits = "0 1 2 3 4 5 6 7 8 9"
OTP = " "
# length of password can be changed
# by changing value in range
for i in range(4):
OTP += digits[math.floor(random.random()*10)]
return OTP
# Driver Code
if __name__ == "__main__":
print("OTP of 4 digits:", generateOTP())
```

NAME: AKASH COURSE: BCA-6-A ROLL No.: 1121007

Q5 Implementing of Encryption and Decryption Using Caeser Cipher

```
=) def encryption(plain_text, key):
        encrypted = " "
    for c in plain_text:
        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
    elif c.islower():
      C_index = ord(c) - ord('a')
    C_shifted = (C_index + key) % 26 + ord('a')
    C_new = chr(C_shifted)
    encrypted += C_new
```

```
elif 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')
c_og = chr(c_og_pos)
    decrypted += c_og
elif c.islower():
    c_index = ord(c) - ord('a')
    c_og_pos = (c_index - key) % 26 + ord('a')
```

```
            c_og = chr(c_og - Pos)
            decrypted += c_og
        elif c.isdigit():
            c_og = (int(c) - key) % 10
            decrypted += str(c_og)
        else
            decrypted += c
    return decrypted
plain_text = "Attack from North"
ciphertext = encryption(plain_text, 4)
Print("Plaintext message:\n", plain_text)
Print("Encrypted ciphertext:\n", ciphertext)
decryptedmsg = decryption(ciphertext, 4)
Print("The decrypted message is:\n", decryptedmsg)
```