

Name: Akash - Bhandari

①

Roll no: 1121008 / BCA '6th A'

MCA Answer

Ans-1 Asymmetric key encryption with Sender's 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

2

```
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 (encryptn - text))
```

```
def encryption (string, key):
```

```
    encrypt - text = []
```

```
    for i in 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 encryption (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 = generate key~~

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__":
```

```
    print("OTP of 4 digit is", generateotp())
```

Akash

Ans-5 Encryption and decryption using caesar 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_pos = (c_index - Key) % 26 + ord('A')
```

Aakash

```
C.os = 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"
```

```
Cipher-text = encryption (Plain-text, 4)
```

```
print (" Plain-text message: \n", Plain-text)
```

```
print (" Encrypted Cipher-text: \n", Cipher-text)
```

```
decrypted msg = decryption (Cipher-text, 4)
```

```
print ("The decrypted message is: \n", decryption)
```

Akash