Name - Naveen Singh

Roll no - 1171087

Section - B

MCQ

1. Asymmetric key encryption with sender public key
3. C. Signature
3. a. Encrypted signature of a sender
4. Cyber laws
5. d. Only ~~a~~ ~~text~~ ~~data~~ alphanumeric
6. Both is same title is different
7. a. Hash value
8. Option a & ~~b~~ C are right.
9. both b and C
10. Possibility of replacement

Name- Naveen Singh
Roll no. - 1171087
Section - B

1.

(i) Prevents unauthorized access to your account

Advanced protection requires security keys for sign in to help protect your google data, like emails, documents, contacts, or other personal google data. Even if a hacker has your username and password, they can't sign in without your security key.

(ii) Provides extra protection from harmful downloads

Advanced protection performs extra checks on downloads, when downloading a file that may be harmful, it notifies you or blocks the download. On your Android phone,

Only apps from verified stores are allowed.

(iii) Keeps your personal information secure

To prevent unauthorized access, Advanced protection only allows google apps and verified third-party apps to access your google account data, and only with your permission.

Advance protection also blocks hackers from impersonating you to access your account: if anyone tries to recover your account, Advance protection also takes extra steps to verify your identity.

Name- Naveen Singh

Rollno - 1121087

4.

```
import math, random
def generateOTP();
digits = "0123456789"
OTP = ""

for i in range(4);
  OTP += digits[math.floor(random.random()*10)]

return OTP
if--name-- == "__main__";
Print("OTP of 4 digits:", generateOTP())
```

5.

Encrypt string Caser cipher:

```
def encrypt(string):
    cipher = ""
    for char in string:
        if char == ' ':
            cipher = cipher + char
        elif char.isupper():
            cipher = cipher + chr((ord(char)+3-65)%26+65)
        else:
            cipher = cipher + chr((ord(char)+3-97)%26+97)

    return cipher
text = "Attack from North"
print("after encryption:", encrypt(text))
```

Decryption string Caser cipher

```
def decrypt(string):
    plain = ""
```

```
for char in string:
  if char = = ():
  plain = plain + char
  elif char.isupper():
  plain = plain + chr(ord(char)-3-65)%26+65)

else:
plain = plain + chr((ord(char)-3-97)%26+97)

return plain

← text = "                    "

← print ("cd after decryption:", decryption(text))
```

Scanned with CamScanner