EX.NO: 01

DATE:

CEASER CIPHER USING PYTHON

AIM:

To write substitution program with encryption and decryption in Ceaser cipher .

ALGORITHM:

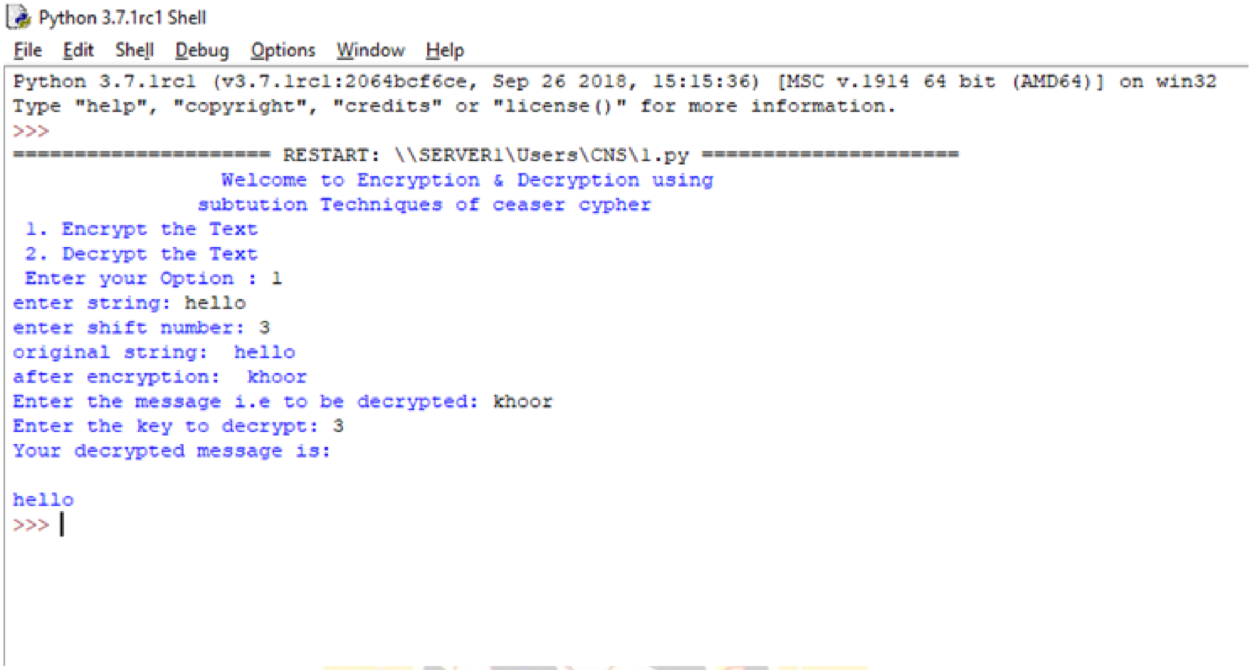
1. Start the program.
2. Read the key value from the user.
3. If the key is positive encrypt the text by adding the key with each character in the plain text.
4. Else subtract the key from the plaintext.
5. Stop the program

PROGRAM:

```
def encrypt(string, shift):


  cipher = ''
  for char in string:
    if char == ' ':
      cipher = cipher + char
    elif  char.isupper():
      cipher = cipher + chr((ord(char) + shift - 65) % 26 + 65)
    else:
      cipher = cipher + chr((ord(char) + shift - 97) % 26 + 97)

  return cipher

print (''' \t\t Welcome to Encryption & Decryption using
               subtution Techniques of ceaser cypher ''' );
print (" 1. Encrypt the Text ");
print (" 2. Decrypt the Text ");

option=int(input(" Enter your Option : "));
if (option == 1):
           text = input("enter string: ");
           s = int(input("enter shift number: "))
           print("original string: ", text)
           print("after encryption: ", encrypt(text, s))
```

```
else:
    print (" Ready To Decrypt  ")

def decrypt():

    #enter your encrypted message(string) below
    encrypted_message = input("Enter the message i.e to be decrypted: ").strip()

    letters="abcdefghijklmnopqrstuvwxyz"

    #enter the key value to decrypt
    k = int(input("Enter the key to decrypt: "))
    decrypted_message = ""

    for ch in encrypted_message:
   if ch in letters:
            position = letters.find(ch)
            new_pos = (position - k) % 26
            new_char = letters[new_pos]
            decrypted_message += new_char
        else:
            decrypted_message += ch
    print("Your decrypted message is:\n")
    print(decrypted_message)
decrypt()
```

OUTPUT:



```
Python 3.7.1rc1 (v3.7.1rc1:2064bcf6ce, Sep 26 2018, 15:15:36) [MSC v.1914 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: \\SERVER1\Users\CNS\1.py =====================
                 Welcome to Encryption & Decryption using
               subtution Techniques of ceaser cypher
 1. Encrypt the Text
 2. Decrypt the Text
 Enter your Option : 1
enter string: hello
enter shift number: 3
original string:  hello
after encryption:  khoor
Enter the message i.e to be decrypted: khoor
Enter the key to decrypt: 3
Your decrypted message is:

hello
>>> 
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:02 **DATE:**	**PLAYFAIR CIPHER USING PYTHON**

AIM:

To write a program to perform encryption and decryption using the substitution techniques of the play-fair in python.

ALGORITHM:

1. Start the program.
2. Declare the variables and assign the values by the matrix.
3. Create the key table for the values by the matrix method.
4. For the encryption using for loop and if statements, two plaintext letters that fall in the same row and matrix are each replaced by the letter to the right, first element of the row move to the last.
5. For the decryption using the following steps use of 5*5 matrix of letters constructed using a keyword.
6. Print the result.
7. Stop the program.

PROGRAM:

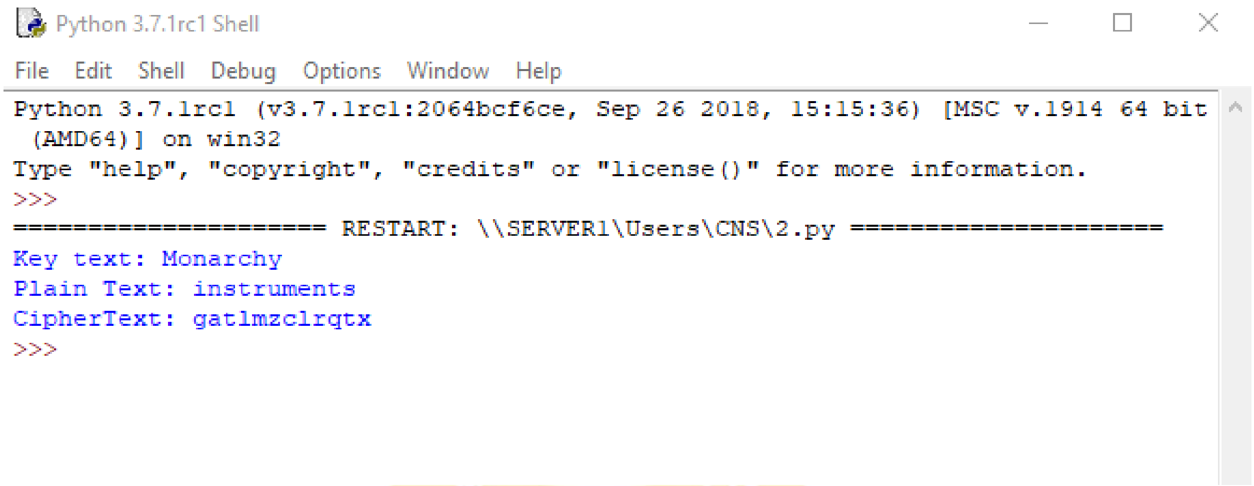
```
def toLowerCase(text):
        return text.lower()


    def removeSpaces(text):
        newText = ""
        for i in text:
            if i == " ":
                continue
            else:
                newText = newText + i
        return newText



    def Diagraph(text):
        Diagraph = []
        group = 0
        for i in range(2, len(text), 2):
            Diagraph.append(text[group:i])

            group = i
        Diagraph.append(text[group:])
        return Diagraph


    def FillerLetter(text):
```

```
    k = len(text)
    if k % 2 == 0:
        for i in range(0, k, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    else:
        for i in range(0, k-1, 2):
            if text[i] == text[i+1]:
                new_word = text[0:i+1] + str('x') + text[i+1:]
                new_word = FillerLetter(new_word)
                break
            else:
                new_word = text
    return new_word


list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',
        'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']


def generateKeyTable(word, list1):
    key_letters = []
    for i in word:
        if i not in key_letters:
            key_letters.append(i)

    compElements = []
    for i in key_letters:
        if i not in compElements:
            compElements.append(i)
    for i in list1:
        if i not in compElements:
            compElements.append(i)

    matrix = []
    while compElements != []:
        matrix.append(compElements[:5])
        compElements = compElements[5:]

    return matrix


def search(mat, element):
    for i in range(5):
        for j in range(5):
            if(mat[i][j] == element):
                return i, j


def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 4:
        char1 = matr[e1r][0]
    else:
        char1 = matr[e1r][e1c+1]

    char2 = ''
    if e2c == 4:
        char2 = matr[e2r][0]
    else:
        char2 = matr[e2r][e2c+1]

    return char1, char2



def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 4:
        char1 = matr[0][e1c]
    else:
        char1 = matr[e1r+1][e1c]
    char2 = "
    if e2r == 4:
        char2 = matr[0][e2c]
    else:
        char2 = matr[e2r+1][e2c]

    return char1, char2



def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = "
    char1 = matr[e1r][e2c]

    char2 = "
    char2 = matr[e2r][e1c]

    return char1, char2



def encryptByPlayfairCipher(Matrix, plainList):
    CipherText = []
    for i in range(0, len(plainList)):
        c1 = 0
        c2 = 0
```

```
            ele1_x, ele1_y = search(Matrix, plainList[i][0])
            ele2_x, ele2_y = search(Matrix, plainList[i][1])

            if ele1_x == ele2_x:
                c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
                # Get 2 letter cipherText
            elif ele1_y == ele2_y:
                c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
            else:
                c1, c2 = encrypt_RectangleRule(
                    Matrix, ele1_x, ele1_y, ele2_x, ele2_y)

            cipher = c1 + c2
            CipherText.append(cipher)
        return CipherText


    text_Plain = 'instruments'
    text_Plain = removeSpaces(toLowerCase(text_Plain))
    PlainTextList = Diagraph(FillerLetter(text_Plain))
    if len(PlainTextList[-1]) != 2:
        PlainTextList[-1] = PlainTextList[-1]+'z'
    key = "Monarchy"
```

```
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)
CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""
for i in CipherList:
        CipherText += i
print("CipherText:", CipherText)
```

OUTPUT:



```
Python 3.7.1rc1 (v3.7.1rc1:2064bcf6ce, Sep 26 2018, 15:15:36) [MSC v.1914 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: \\SERVER1\Users\CNS\2.py =====================
Key text: Monarchy
Plain Text: instruments
CipherText: gatlmzclrqtx
>>>
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:03

DATE:

HILL CIPHER USING PYTHON

AIM:

To write a program for encryption and decryption using substitution techniques of hill cipher in python language.

ALGORITHM:

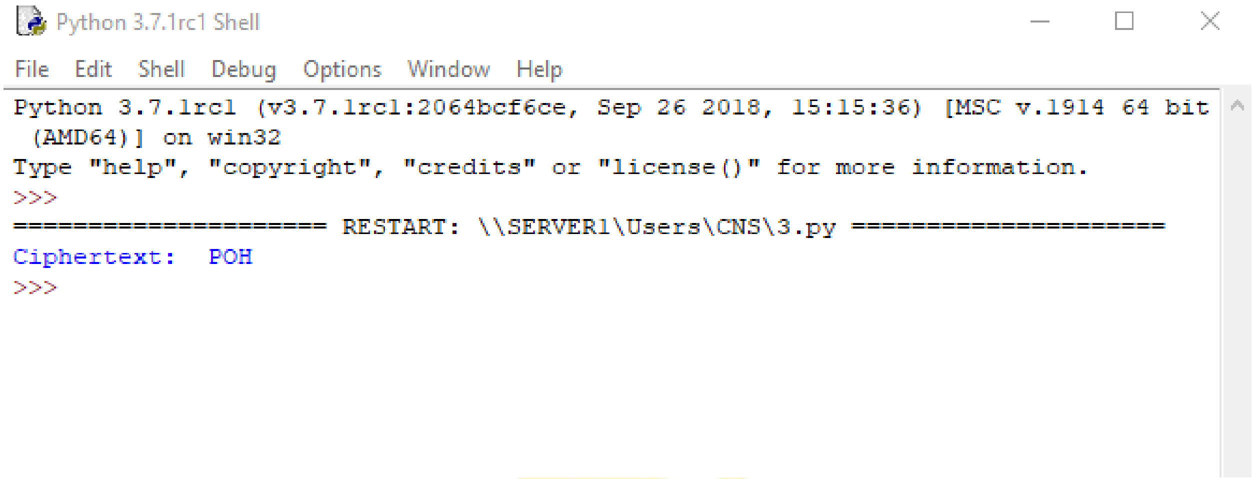
1. Start the program.
2. Declare the variable.
3. Generate the key matrix and cipher matrix for the encryption.
4. Using for loop to get the key matrix for the key.
5. Using for loop to get the range and stored on the cipher matrix.
6. By getting stored cipher matrix to get the encrypted message and key.
7. For the decryption, by reversing the method to get the original message.
8. Print the result.

PROGRAM:

```
keyMatrix = [[0] * 3 for i in range(3)]

messageVector = [[0] for i in range(3)]

cipherMatrix = [[0] for i in range(3)]

def getKeyMatrix(key):
    k = 0
    for i in range(3):
        for j in range(3):
            keyMatrix[i][j] = ord(key[k]) % 65
            k += 1

def encrypt(messageVector):
    for i in range(3):
        for j in range(1):
            cipherMatrix[i][j] = 0
            for x in range(3):
                cipherMatrix[i][j] += (keyMatrix[i][x] *
                                                  messageVector[x][j])
            cipherMatrix[i][j] = cipherMatrix[i][j] % 26
```

```
def HillCipher(message, key):

    getKeyMatrix(key)

    for i in range(3):
        messageVector[i][0] = ord(message[i]) % 65

    encrypt(messageVector)

    CipherText = []
    for i in range(3):
        CipherText.append(chr(cipherMatrix[i][0] + 65))
    print("Ciphertext: ", "".join(CipherText))

def main():
    message = "ACT"
    key = "GYBNQKURP"
    HillCipher(message, key)
if __name__ == "__main__":
    main()
```

OUTPUT:



```
Python 3.7.1rc1 (v3.7.1rc1:2064bcf6ce, Sep 26 2018, 15:15:36) [MSC v.1914 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: \\SERVER1\Users\CNS\3.py =====================
Ciphertext:  POH
>>>
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:04 DATE:	VIGENERE CIPHER USING PYTHON

AIM:

To write a program for encryption and decryption using substitution techniques of Vigenere cipher in python language.

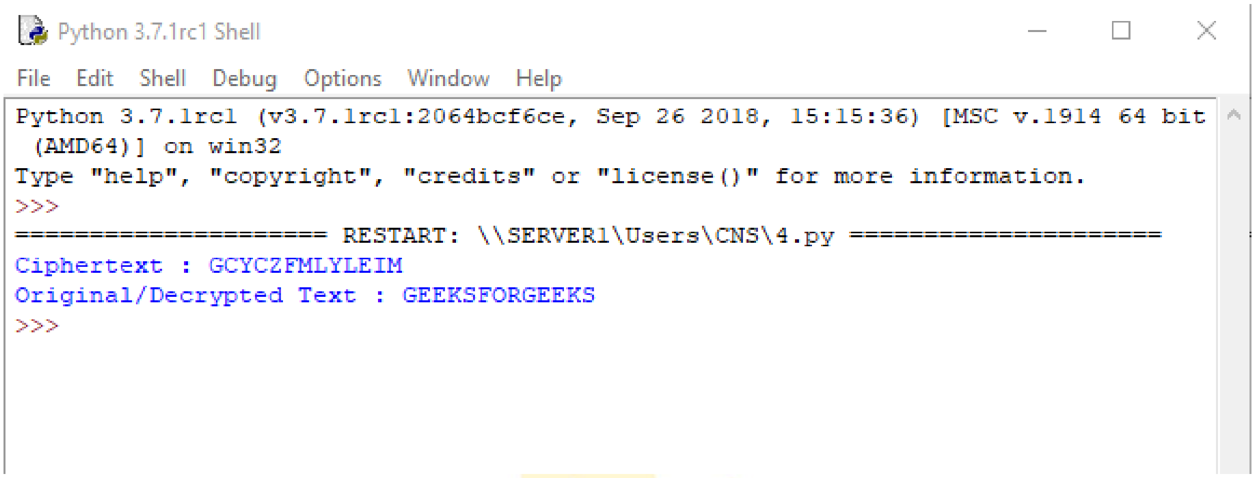
ALGORITHM:

1. Start the program
2. Declare the variables
3. Generate the key by the length of the original text.
4. For the encryption using for loop to encrypt/decrypt the text one by one.
5. Get the cipher text by using for loop to check one by one and compare with key.
6. Print the result.
7. Stop the program.

PROGRAM:

```
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(key))


def cipherText(string, key):

    cipher_text = []

    for i in range(len(string)):

        x = (ord(string[i]) +

            ord(key[i])) % 26

        x += ord('A')

        cipher_text.append(chr(x))

    return("" . join(cipher_text))


def originalText(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))


if __name__ == "__main__":
        string = "GEEKSFORGEEKS"
        keyword = "AYUSH"
        key = generateKey(string, keyword)
        cipher_text = cipherText(string,key)
        print("Ciphertext :", cipher_text)
        print("Original/Decrypted Text :",
                originalText(cipher_text, key))
```

OUTPUT:



```
Python 3.7.1rc1 (v3.7.1rc1:2064bcf6ce, Sep 26 2018, 15:15:36) [MSC v.1914 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===================== RESTART: \\SERVER1\Users\CNS\4.py =====================
Ciphertext : GCYCZFMLYLEIM
Original/Decrypted Text : GEEKSFORGEEKS
>>>
```

RESULT:

Thus the above program is verified and executed successfully..

EX.NO:05 DATE:	RAIL FENCE – ROW AND COLUMN TRANSFORMATION

AIM:

To write a program encryption and decryption using the transposition techniques Rail fence- rows and column transformation in python.

ALGORITHM:

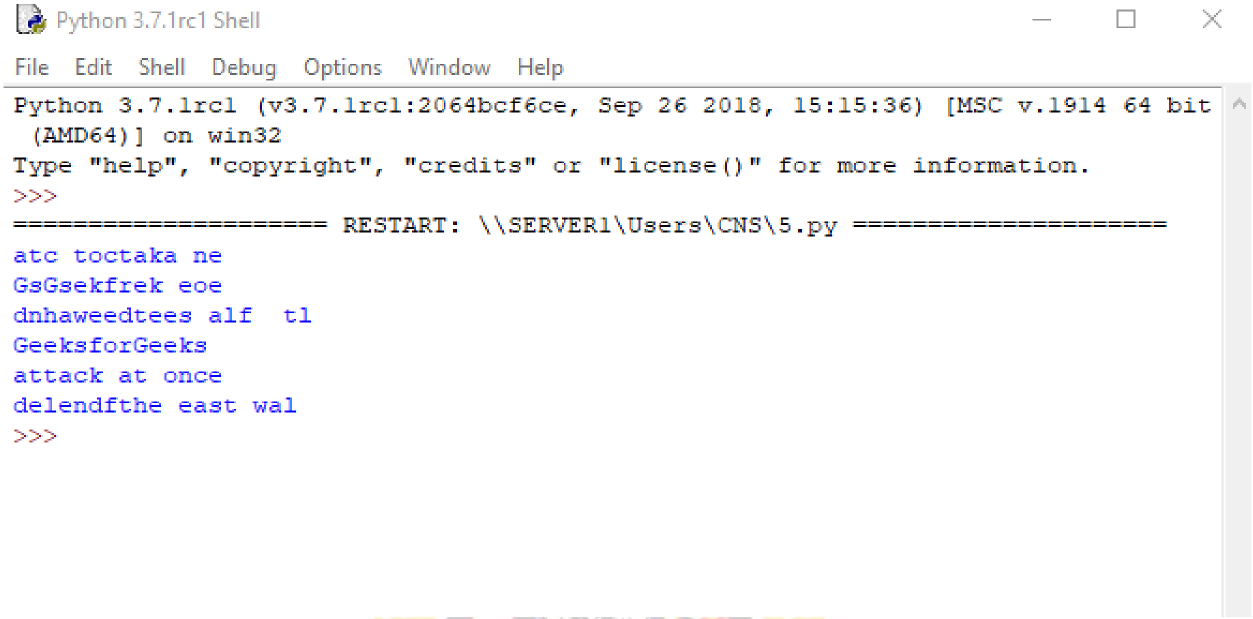
1. Start the program.
2. Declare the class and required variables.
3. Declare the constructor of the class.
4. Implement the Rail fence cipher.
5. Access the member functions using the objects.
6. Display the result.
7. Stop the program.

PROGRAM:

```
def encryptRailFence(text, key):
    rail = [['\n' for i in range(len(text))]
                  for j in range(key)]

    dir_down = False
    row, col = 0, 0

    for i in range(len(text)):

        if (row == 0) or (row == key - 1):
            dir_down = not dir_down

        rail[row][col] = text[i]
        col += 1

        if dir_down:
            row += 1
        else:
            row -= 1
    result = []
    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':

                result.append(rail[i][j])

    return("" . join(result))


def decryptRailFence(cipher, key):


    rail = [['\n' for i in range(len(cipher))]

                for j in range(key)]


    dir_down = None

    row, col = 0, 0


    for i in range(len(cipher)):

        if row == 0:

            dir_down = True

        if row == key - 1:

            dir_down = False


        rail[row][col] = '*'

        col += 1


        if dir_down:

            row += 1
        else:
            row -= 1

    index = 0
    for i in range(key):
        for j in range(len(cipher)):
            if ((rail[i][j] == '*') and
            (index < len(cipher))):
                rail[i][j] = cipher[index]
                index += 1

    result = []
    row, col = 0, 0
    for i in range(len(cipher)):


        if row == 0:
            dir_down = True
        if row == key-1:
            dir_down = False

        if (rail[row][col] != '*'):
            result.append(rail[row][col])
```

```
            col += 1

        if dir_down:
            row += 1
        else:
            row -= 1
    return("".join(result))

if __name__ == "__main__":
    print(encryptRailFence("attack at once", 2))
    print(encryptRailFence("GeeksforGeeks ", 3))
    print(encryptRailFence("defend the east wall", 3))
    print(decryptRailFence("GsGsekfrek eoe", 3))
    print(decryptRailFence("atc toctaka ne", 2))
    print(decryptRailFence("dnhaweedtees alf tl", 3))
```

OUTPUT:



```
Python 3.7.1rc1 (v3.7.1rc1:2064bcf6ce, Sep 26 2018, 15:15:36) [MSC v.1914 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
===================== RESTART: \\SERVER1\Users\CNS\5.py =====================
atc toctaka ne
GsGsekfrek eoe
dnhaweedtees alf  tl
GeeksforGeeks
attack at once
delendfthe east wal
>>> 
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:06

DATE:

DATA ENCRYPTION STANDARD USING PYTHON

AIM:

To write a python program to implement DES using python programming.

ALGORITHM:

1. Start the program.
2. Read the 64 bit plane text.
3. Split it into two 32-bit blocks and store it in two different arrays.
4. Perform XOR operation between these two arrays.
5. The output obtained is stored as the second 32-bit sequence and the original second 32-bit sequence forms the first part.
6. Thus the encryption 64 bit cipher text is obtained in this way.
7. Repeat the same process for the remaining plain text characters.
8. Stop the program.

PROGRAM:

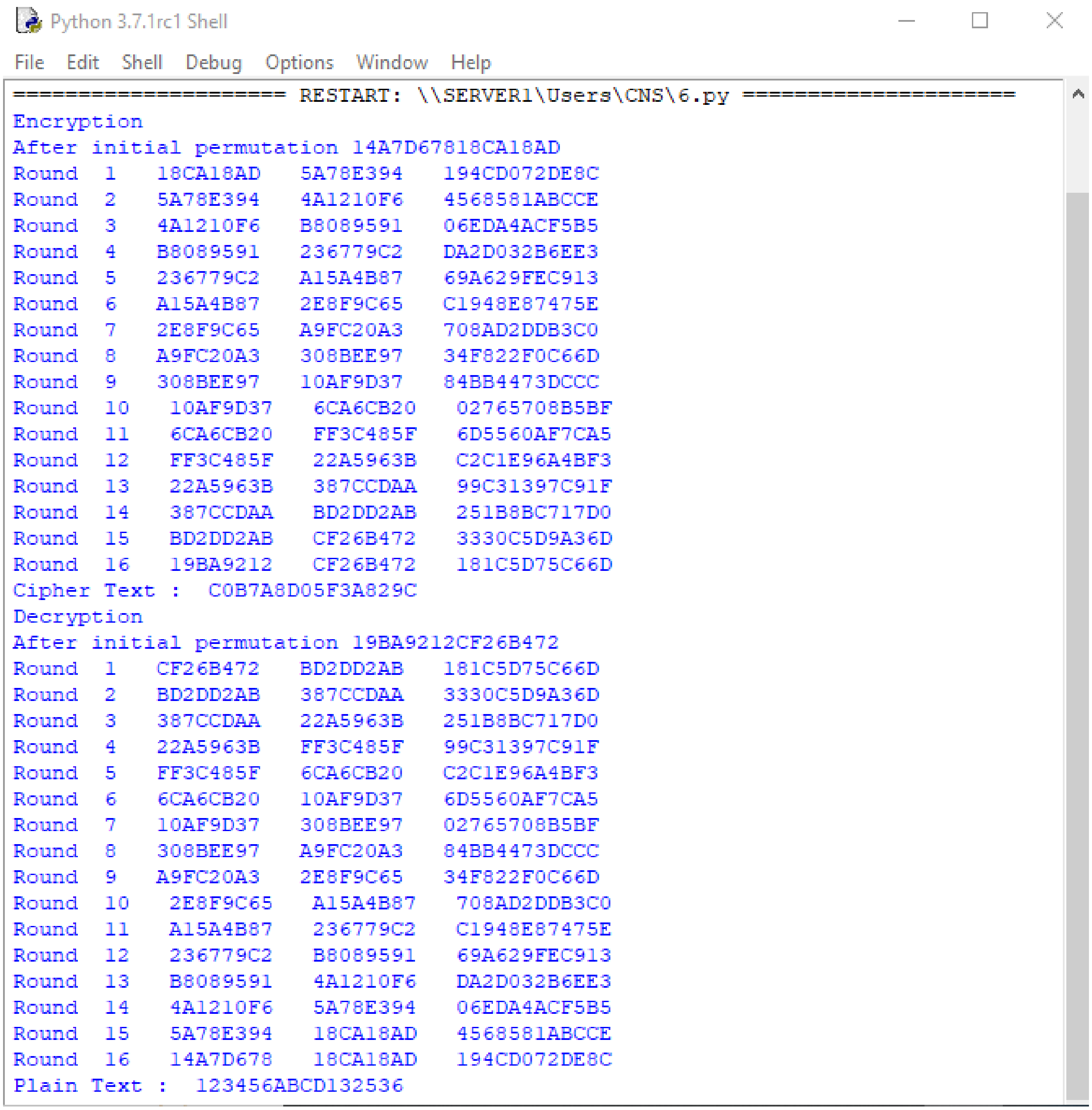
```
def hex2bin(s):

        mp = {'0': "0000",

                '1': "0001",

                '2': "0010",

                '3': "0011",

                '4': "0100",

                '5': "0101",

                '6': "0110",

                '7': "0111",

                '8': "1000",

                '9': "1001",

                'A': "1010",

                'B': "1011",

                'C': "1100",

                'D': "1101",

                'E': "1110",

                'F': "1111"}

        bin = ""

        for i in range(len(s)):

                bin = bin + mp[s[i]]

        return bin
```

```
def bin2hex(s):
        mp = {"0000": '0',
                "0001": '1',
                "0010": '2',
                "0011": '3',
                "0100": '4',
                "0101": '5',
                "0110": '6',
                "0111": '7',
                "1000": '8',
                "1001": '9',
                "1010": 'A',
                "1011": 'B',
                "1100": 'C',
                "1101": 'D',
                "1110": 'E',
                "1111": 'F'}
        hex = ""
        for i in range(0, len(s), 4):
                ch = ""
                ch = ch + s[i]
                ch = ch + s[i + 1]
                ch = ch + s[i + 2]
        ch = ch + s[i + 3]

        hex = hex + mp[ch]


    return hex
def bin2dec(binary):


    binary1 = binary

    decimal, i, n = 0, 0, 0

    while(binary != 0):

        dec = binary % 10

        decimal = decimal + dec * pow(2, i)

        binary = binary//10

        i += 1

    return decimal


def dec2bin(num):

    res = bin(num).replace("0b", "")

    if(len(res) % 4 != 0):

        div = len(res) / 4

        div = int(div)

        counter = (4 * (div + 1)) - len(res)

        for i in range(0, counter):

            res = '0' + res
    return res



def permute(k, arr, n):
    permutation = ""
    for i in range(0, n):
        permutation = permutation + k[arr[i] - 1]
    return permutation
def shift_left(k, nth_shifts):
    s = ""
    for i in range(nth_shifts):
        for j in range(1, len(k)):
            s = s + k[j]
        s = s + k[0]
        k = s
        s = ""
    return k
def xor(a, b):
    ans = ""
    for i in range(len(a)):
        if a[i] == b[i]:
            ans = ans + "0"
        else:
            ans = ans + "1"
    return ans

initial_perm = [58, 50, 42, 34, 26, 18, 10, 2,
                60, 52, 44, 36, 28, 20, 12, 4,
                62, 54, 46, 38, 30, 22, 14, 6,
                64, 56, 48, 40, 32, 24, 16, 8,
                57, 49, 41, 33, 25, 17, 9, 1,
                59, 51, 43, 35, 27, 19, 11, 3,
                61, 53, 45, 37, 29, 21, 13, 5,
                63, 55, 47, 39, 31, 23, 15, 7]

exp_d = [32, 1, 2, 3, 4, 5, 4, 5,
        6, 7, 8, 9, 8, 9, 10, 11,
        12, 13, 12, 13, 14, 15, 16, 17,
        16, 17, 18, 19, 20, 21, 20, 21,
        22, 23, 24, 25, 24, 25, 26, 27,
        28, 29, 28, 29, 30, 31, 32, 1]

per = [16, 7, 20, 21,
    29, 12, 28, 17,
    1, 15, 23, 26,
    5, 18, 31, 10,
        2, 8, 24, 14,
        32, 27, 3, 9,
        19, 13, 30, 6,
        22, 11, 4, 25]

sbox = [[[14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7],
         [0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8],
         [4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0],
         [15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13]],

         [[15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10],
         [3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5],
         [0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15],
         [13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9]],

         [[10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8],
         [13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1],
         [13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7],
         [1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12]],

         [[7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15],
         [13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9],
         [10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4],
         [3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14]],

         [[2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9],
         [14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6],
         [4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14],
         [11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3]],

         [[12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11],
         [10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8],
         [9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6],
         [4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13]],

         [[4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1],
         [13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6],
         [1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2],
         [6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12]],

         [[13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7],
         [1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2],
         [7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8],
         [2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11]]]
```

```
# Final Permutation Table
final_perm = [40, 8, 48, 16, 56, 24, 64, 32,
              39, 7, 47, 15, 55, 23, 63, 31,
              38, 6, 46, 14, 54, 22, 62, 30,
              37, 5, 45, 13, 53, 21, 61, 29,
              36, 4, 44, 12, 52, 20, 60, 28,
              35, 3, 43, 11, 51, 19, 59, 27,
              34, 2, 42, 10, 50, 18, 58, 26,
              33, 1, 41, 9, 49, 17, 57, 25]


def encrypt(pt, rkb, rk):
    pt = hex2bin(pt)

    # Initial Permutation
    pt = permute(pt, initial_perm, 64)
    print("After initial permutation", bin2hex(pt))

    # Splitting
    left = pt[0:32]
    right = pt[32:64]
    for i in range(0, 16):
        # Expansion D-box: Expanding the 32 bits data into 48 bits
        right_expanded = permute(right, exp_d, 48)


        # XOR RoundKey[i] and right_expanded
        xor_x = xor(right_expanded, rkb[i])


        # S-boxex: substituting the value from s-box table by calculating row
and column
        sbox_str = ""
        for j in range(0, 8):
            row = bin2dec(int(xor_x[j * 6] + xor_x[j * 6 + 5]))
            col = bin2dec(
                int(xor_x[j * 6 + 1] + xor_x[j * 6 + 2] + xor_x[j * 6 + 3]
+ xor_x[j * 6 + 4]))
            val = sbox[j][row][col]
            sbox_str = sbox_str + dec2bin(val)


        # Straight D-box: After substituting rearranging the bits
        sbox_str = permute(sbox_str, per, 32)


        # XOR left and sbox_str
        result = xor(left, sbox_str)
        left = result
        # Swapper
        if(i != 15):
            left, right = right, left
        print("Round ", i + 1, " ", bin2hex(left),
            " ", bin2hex(right), " ", rk[i])

    # Combination
    combine = left + right

    # Final permutation: final rearranging of bits to get cipher text
    cipher_text = permute(combine, final_perm, 64)
    return cipher_text


pt = "123456ABCD132536"
key = "AABB09182736CCDD"

key = hex2bin(key)

# --parity bit drop table
keyp = [57, 49, 41, 33, 25, 17, 9,
        1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27,
                19, 11, 3, 60, 52, 44, 36,
                63, 55, 47, 39, 31, 23, 15,
                7, 62, 54, 46, 38, 30, 22,
                14, 6, 61, 53, 45, 37, 29,
                21, 13, 5, 28, 20, 12, 4]

# getting 56 bit key from 64 bit using the parity bits
key = permute(key, keyp, 56)

# Number of bit shifts
shift_table = [1, 1, 2, 2,
                    2, 2, 2, 2,
                    1, 2, 2, 2,
                    2, 2, 2, 1]

# Key- Compression Table : Compression of key from 56 bits to 48 bits
key_comp = [14, 17, 11, 24, 1, 5,
                    3, 28, 15, 6, 21, 10,
                    23, 19, 12, 4, 26, 8,
                    16, 7, 27, 20, 13, 2,
                    41, 52, 31, 37, 47, 55,
                    30, 40, 51, 45, 33, 48,
                    44, 49, 39, 56, 34, 53,
                        46, 42, 50, 36, 29, 32]


# Splitting
left = key[0:28] # rkb for RoundKeys in binary
right = key[28:56] # rk for RoundKeys in hexadecimal


rkb = []
rk = []
for i in range(0, 16):
    # Shifting the bits by nth shifts by checking from shift table
    left = shift_left(left, shift_table[i])
    right = shift_left(right, shift_table[i])


    # Combination of left and right string
    combine_str = left + right


    # Compression of key from 56 to 48 bits
    round_key = permute(combine_str, key_comp, 48)


    rkb.append(round_key)
    rk.append(bin2hex(round_key))


print("Encryption")
```

```
cipher_text = bin2hex(encrypt(pt, rkb, rk))
print("Cipher Text : ", cipher_text)


print("Decryption")
rkb_rev = rkb[::-1]
rk_rev = rk[::-1]
text = bin2hex(encrypt(cipher_text, rkb_rev, rk_rev))
print("Plain Text : ", text)
```

OUTPUT:



```
===================== RESTART: \\SERVER1\Users\CNS\6.py =====================
Encryption
After initial permutation 14A7D67818CA18AD
Round  1   18CA18AD   5A78E394   194CD072DE8C
Round  2   5A78E394   4A1210F6   4568581ABCCE
Round  3   4A1210F6   B8089591   06EDA4ACF5B5
Round  4   B8089591   236779C2   DA2D032B6EE3
Round  5   236779C2   A15A4B87   69A629FEC913
Round  6   A15A4B87   2E8F9C65   C1948E87475E
Round  7   2E8F9C65   A9FC20A3   708AD2DDB3C0
Round  8   A9FC20A3   308BEE97   34F822F0C66D
Round  9   308BEE97   10AF9D37   84BB4473DCCC
Round  10   10AF9D37   6CA6CB20   02765708B5BF
Round  11   6CA6CB20   FF3C485F   6D5560AF7CA5
Round  12   FF3C485F   22A5963B   C2C1E96A4BF3
Round  13   22A5963B   387CCDAA   99C31397C91F
Round  14   387CCDAA   BD2DD2AB   251B8BC717D0
Round  15   BD2DD2AB   CF26B472   3330C5D9A36D
Round  16   19BA9212   CF26B472   181C5D75C66D
Cipher Text :  C0B7A8D05F3A829C
Decryption
After initial permutation 19BA9212CF26B472
Round  1   CF26B472   BD2DD2AB   181C5D75C66D
Round  2   BD2DD2AB   387CCDAA   3330C5D9A36D
Round  3   387CCDAA   22A5963B   251B8BC717D0
Round  4   22A5963B   FF3C485F   99C31397C91F
Round  5   FF3C485F   6CA6CB20   C2C1E96A4BF3
Round  6   6CA6CB20   10AF9D37   6D5560AF7CA5
Round  7   10AF9D37   308BEE97   02765708B5BF
Round  8   308BEE97   A9FC20A3   84BB4473DCCC
Round  9   A9FC20A3   2E8F9C65   34F822F0C66D
Round  10   2E8F9C65   A15A4B87   708AD2DDB3C0
Round  11   A15A4B87   236779C2   C1948E87475E
Round  12   236779C2   B8089591   69A629FEC913
Round  13   B8089591   4A1210F6   DA2D032B6EE3
Round  14   4A1210F6   5A78E394   06EDA4ACF5B5
Round  15   5A78E394   18CA18AD   4568581ABCCE
Round  16   14A7D678   18CA18AD   194CD072DE8C
Plain Text :  123456ABCD132536
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:07 **DATE:**	**ADVANCED ENCRYPTION STANDARD**

AIM:

To write a program to apply AES algorithm using Python.

ALGORITHM:

1. Start the program.

2. The plain text that encrypted using with the key k1.

3. Des that do the decryption using the key k2.

4. Decryption is the reversed version of Encryption.

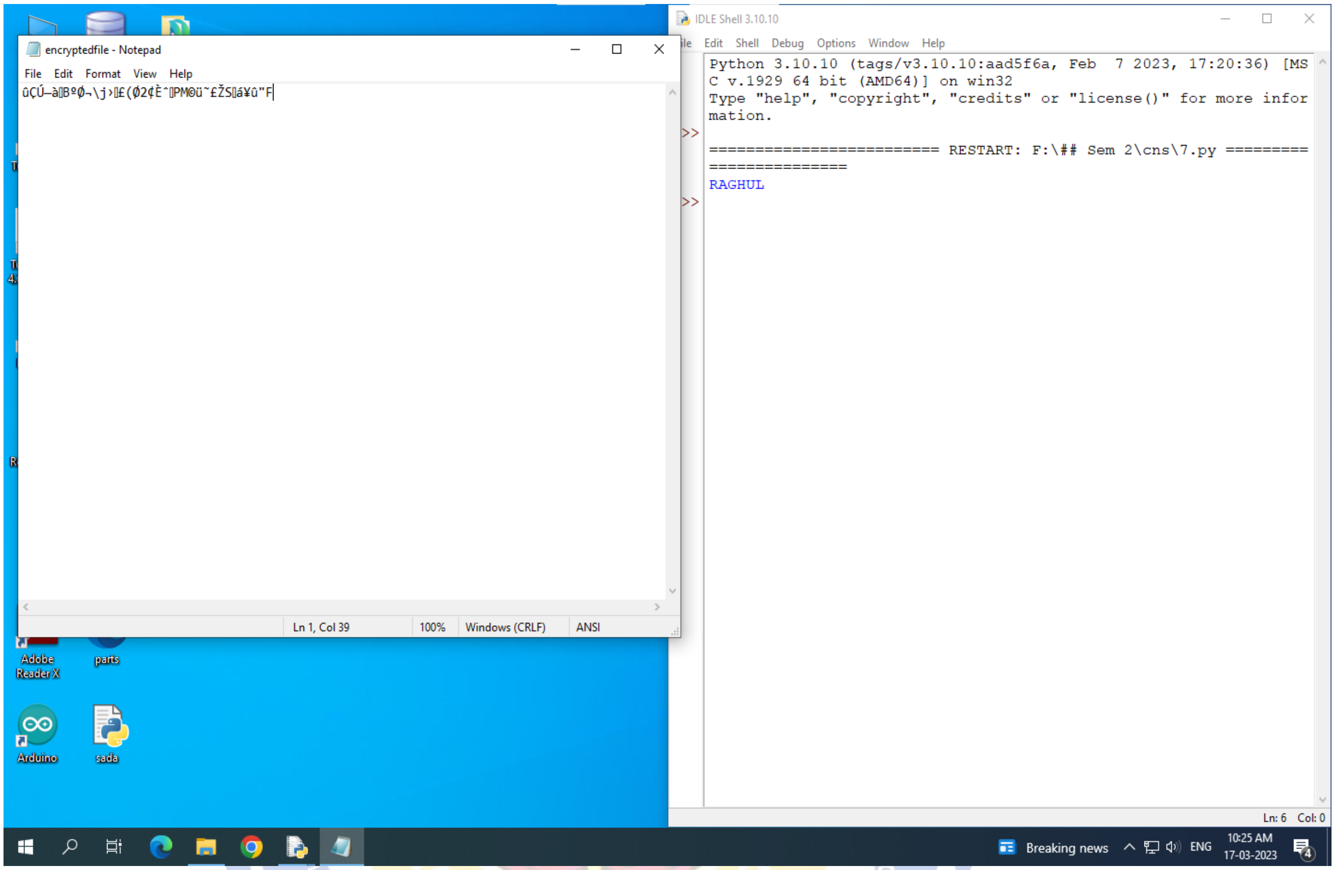
5. Display the output.

6. Stop the process.

PROGRAM:

```
from Cryptodome.Cipher import AES
from Cryptodome.Random import get_random_bytes

#define our data
data=b"SECRETDATA"

key = get_random_bytes(16)
cipher = AES.new(key, AES.MODE_EAX)
ciphertext, tag = cipher.encrypt_and_digest(data)

file_out = open("encryptedfile.bin", "wb")
[ file_out.write(x) for x in (cipher.nonce, tag, ciphertext) ]
file_out.close()

file_in = open("encryptedfile.bin", "rb")
nonce, tag, ciphertext = [ file_in.read(x) for x in (16, 16, -1) ]

#the person decrypting the message will need access to the key
cipher = AES.new(key, AES.MODE_EAX, nonce)
data = cipher.decrypt_and_verify(ciphertext, tag)
print(data.decode('UTF-8'))
```

OUTPUT:



```
ûÇÚ–à▯BºØ¬\j›▯£(Ø2¢È˜▯PM©ü˜£ŽS▯á¥û"F
```

```
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MS
C v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more infor
mation.
>>
========================= RESTART: F:\## Sem 2\cns\7.py =========
===============
RAGHUL
>>
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:08 DATE:	TRIPLE DES

AIM:

To write a program to implement the Triple DES in Java.

ALGORITHM:

1. Start the program.
2. Triple DES contains three keys k1, k2 and k3, each of 56 bits.
3. DES encrypt with k1.
4. DES encrypt with k2.
5. DES encrypt with k3.
6. Decryption is the reverse of encryption, decrypt with k3, encrypt with k2, then decrypt with k1.
7. Display the output.
8. Stop the program.

PROGRAM:

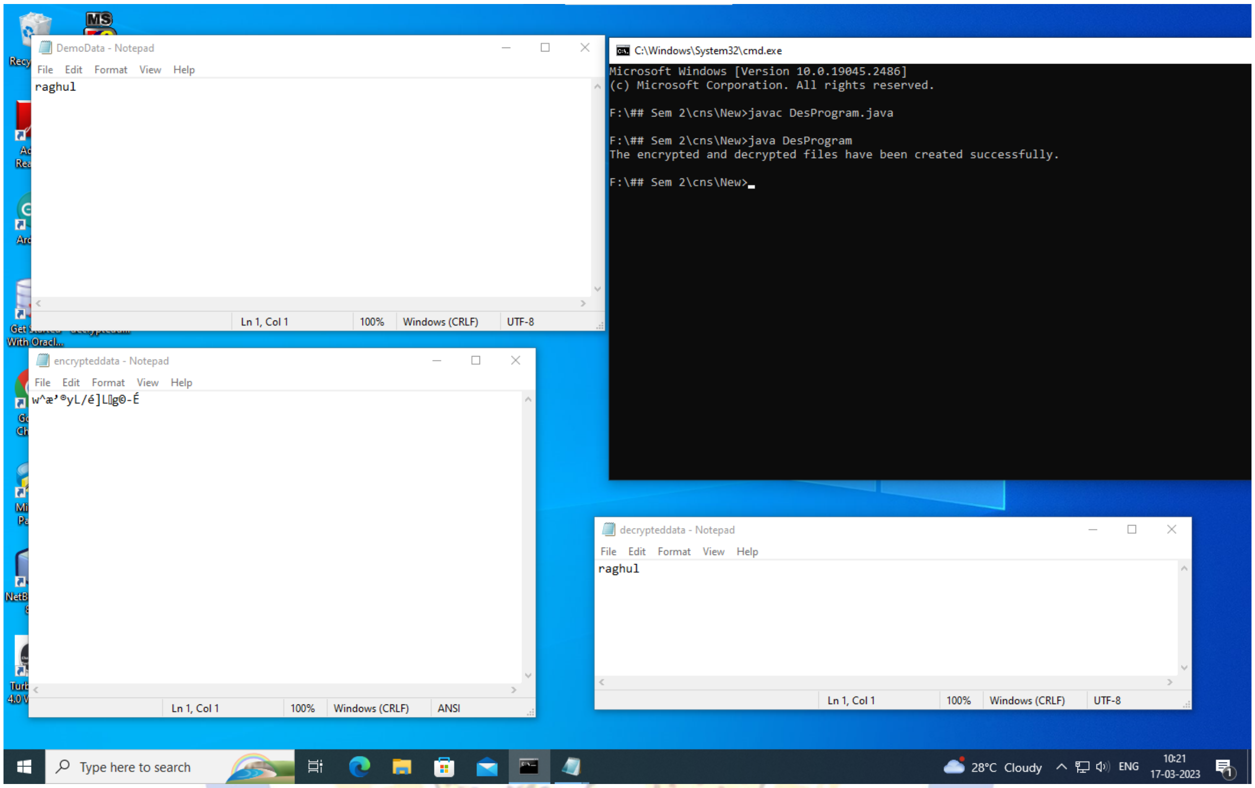
```
//Java classes are mandatory to import for encryption and decryption process
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.security.InvalidAlgorithmParameterException;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
public class DesProgram
{
//creating an instance of the Cipher class for encryption
private static Cipher encrypt;
//creating an instance of the Cipher class for decryption
```

```
private static Cipher decrypt;
//initializing vector
private static final byte[] initialization_vector = { 22, 33, 11, 44, 55, 99, 66, 77 };
//main() method
public static void main(String[] args)
{
//path of the file that we want to encrypt
String textFile = "C:/Users/2k22it17/Desktop/DemoData.txt";
//path of the encrypted file that we get as output
String encryptedData = "C:/Users/2k22it17/Desktop/encrypteddata.txt";
//path of the decrypted file that we get as output
String decryptedData = "C:/Users/2k22it17/Desktop/decrypteddata.txt";
try
{
//generating keys by using the KeyGenerator class
SecretKey scrtkey = KeyGenerator.getInstance("DES").generateKey();
AlgorithmParameterSpec aps = new IvParameterSpec(initialization_vector);
//setting encryption mode
encrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
encrypt.init(Cipher.ENCRYPT_MODE, scrtkey, aps);
//setting decryption mode
decrypt = Cipher.getInstance("DES/CBC/PKCS5Padding");
decrypt.init(Cipher.DECRYPT_MODE, scrtkey, aps);
//calling encrypt() method to encrypt the file
encryption(new FileInputStream(textFile), new FileOutputStream(encryptedData));
//calling decrypt() method to decrypt the file
decryption(new FileInputStream(encryptedData), new FileOutputStream(decryptedData));
//prints the stetment if the program runs successfully
System.out.println("The encrypted and decrypted files have been created successfully.");
}
//catching multiple exceptions by using the | (or) operator in a single catch block
catch (NoSuchAlgorithmException | NoSuchPaddingException | InvalidKeyException | InvalidAlgorithmParameterException | IOException e)
{
//prints the message (if any) related to exceptions
e.printStackTrace();
}
}
//method for encryption
private static void encryption(InputStream input, OutputStream output)
throws IOException
{
output = new CipherOutputStream(output, encrypt);
//calling the writeBytes() method to write the encrypted bytes to the file
```

```
writeBytes(input, output);
}
//method for decryption
private static void decryption(InputStream input, OutputStream output)
throws IOException
{
input = new CipherInputStream(input, decrypt);
//calling the writeBytes() method to write the decrypted bytes to the file
writeBytes(input, output);
}
//method for writting bytes to the files
private static void writeBytes(InputStream input, OutputStream output)
throws IOException
{
byte[] writeBuffer = new byte[512];
int readBytes = 0;
while ((readBytes = input.read(writeBuffer)) >= 0)
{
output.write(writeBuffer, 0, readBytes);
}
//closing the output stream
output.close();
//closing the input stream
```

```
input.close();
}
}
```

OUTPUT:



DemoData:

```
raghul
```

encrypteddata:

```
w^æ’®yL/é]L▯g©-É
```

decrypteddata:

```
raghul
```

```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

F:\## Sem 2\cns\New>javac DesProgram.java

F:\## Sem 2\cns\New>java DesProgram
The encrypted and decrypted files have been created successfully.

F:\## Sem 2\cns\New>_
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:09

DATE:

RSA ALGORITHM

AIM:

To write a program to implement RSA algorithm using HTML and JavaScript.

ALGORITHM:

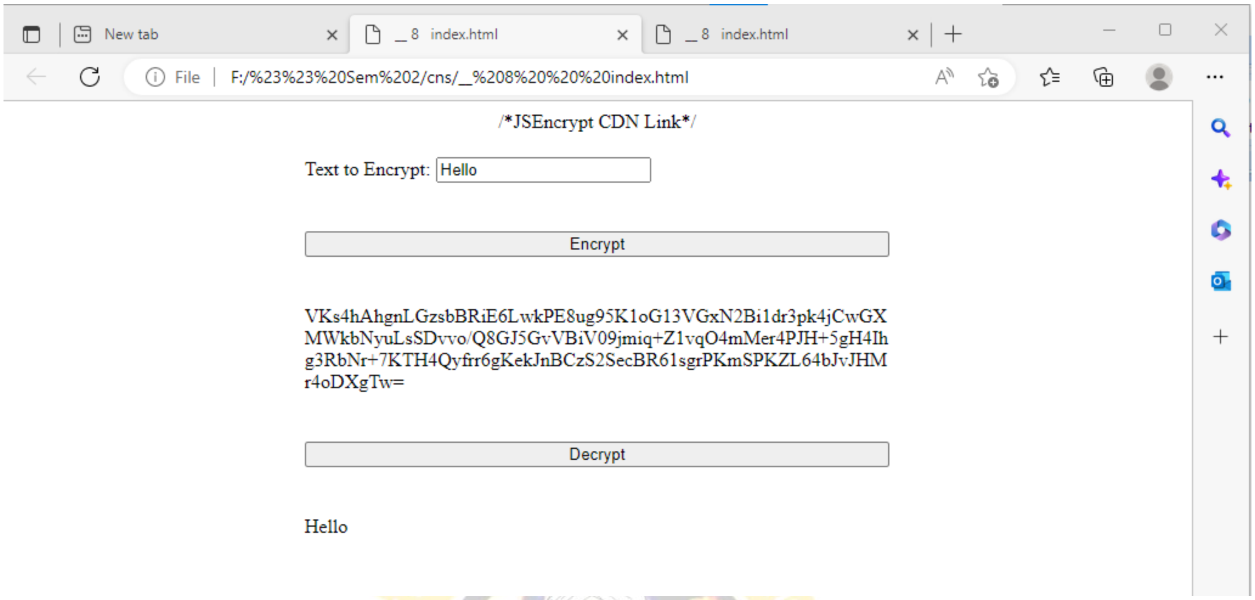
1. Start the program.
2. Declare the variables and values.
3. Generate the public key and private key for the encrypted key.
4. Get the values from user and process by using RSA algorithm.
5. Using JavaScript to encrypt and decrypt the values that user entered.
6. Display the result.
7. Stop the program.

PROGRAM:

```
<html>
<head>
/*JSEncrypt CDN Link*/
<script
src="https://cdnjs.cloudflare.com/ajax/libs/jsencrypt/3.1.0/jsencrypt.min.js"
integrity="sha512-
Tl9i44ZZYtGq56twOViooxyXCSNNkEkRmDMnPAmgU+m8B8A8LXJemzkH/s
Z7y4BWi5kVVfkr75v+CQDU6Ug+yw==" crossorigin="anonymous">
</script>
<script>
/*Creating instance*/
var cryptofunction = new JSEncrypt();
var ciphertext;
var originaltext;
/*Generating public and private key*/
var pubickey = cryptofunction.getPublicKey();
var pvtkey = cryptofunction.getPrivateKey();
/*Setting public and private key*/
cryptofunction.setPublicKey(pubickey);
cryptofunction.setPrivateKey(pvtkey);
/*function to perform encryption*/
function performEncryption() {
var tempval = document.getElementById('inputtext').value;
ciphertext = cryptofunction.encrypt(tempval);
document.getElementById('encrptedtext').innerHTML = ciphertext;
}
/*function to perform decryption*/
function performDecryption() {
originaltext = cryptofunction.decrypt(ciphertext);
document.getElementById('decryptedtext').innerHTML = originaltext;
}
```

```
</script>
<style>
.margins {
margin:20px;
width:50%;
}
#encrptedtext {
word-break: break-all;
}
.parent {
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
}
</style>
</head>
<body class="parent">
<div class="margins">Text to Encrypt: <input type="text" id="inputtext"/></div>
<button class="margins" onclick="performEncryption()">Encrypt</button>
<div class="margins" id="encrptedtext"></div>
<button class="margins" onclick="performDecryption()">Decrypt</button>
<div class="margins" id="decryptedtext"></div>
</body>
</html>
```

OUTPUT:



/*JSEncrypt CDN Link*/

Text to Encrypt: Hello

Encrypt

```
VKs4hAhgnLGzsbBRiE6LwkPE8ug95K1oG13VGxN2Bi1dr3pk4jCwGX
MWkbNyuLsSDvvo/Q8GJ5GvVBiV09jmiq+Z1vqO4mMer4PJH+5gH4Ih
g3RbNr+7KTH4Qyfrr6gKekJnBCzS2SecBR61sgrPKmSPKZL64bJvJHM
r4oDXgTw=
```

Decrypt

Hello

RESULT:

Thus the above program verified and executed successfully.

EX.NO:10 **DATE:**	**DIFFIE-HELLMAN KEY EXCHANGE ALGORITHM**

AIM:

To write a program to implement Diffie - Hellman Key Exchange algorithm for encryption and decryption.

ALGORITHM:

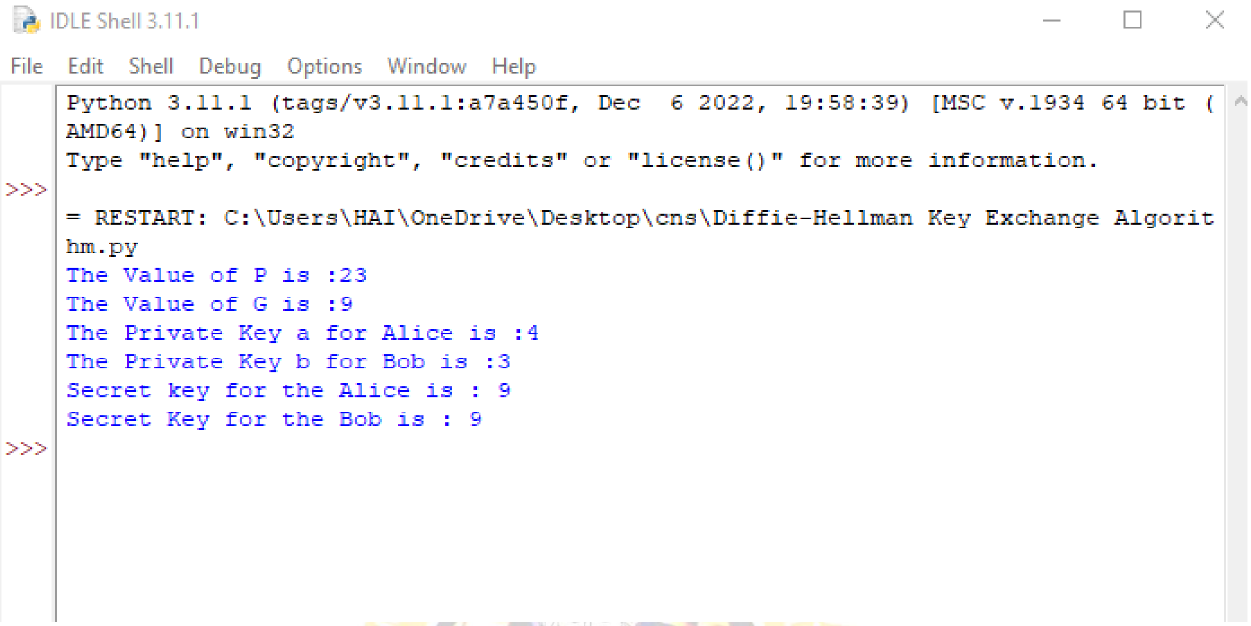
1. Start the program.
2. Declare the variables and values.
3. Generate the html from for user and process by using RSA algorithm.
4. Using Javascript to encrypt and decrypt the values that user entered.
5. Display the result.
6. Stop the process.

PROGRAM:

```
from random import randint

if __name__ == '__main__':

# Both the persons will be agreed upon the
# public keys G and P
# A prime number P is taken
P = 23

# A primitive root for P, G is taken
G = 9
print('The Value of P is :%d'%(P))
print('The Value of G is :%d'%(G))

# Alice will choose the private key a
a = 4
print('The Private Key a for Alice is :%d'%(a))

# gets the generated key
x = int(pow(G,a,P))
```

```
# Bob will choose the private key b

b = 3

print('The Private Key b for Bob is :%d'%(b))


# gets the generated key

y = int(pow(G,b,P))

# Secret key for Alice

ka = int(pow(y,a,P))

# Secret key for Bob

kb = int(pow(x,b,P))


print('Secret key for the Alice is : %d'%(ka))

print('Secret Key for the Bob is : %d'%(kb))
```

OUTPUT:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\HAI\OneDrive\Desktop\cns\Diffie-Hellman Key Exchange Algorit
hm.py
The Value of P is :23
The Value of G is :9
The Private Key a for Alice is :4
The Private Key b for Bob is :3
Secret key for the Alice is : 9
Secret Key for the Bob is : 9
>>>
```

RESULT:

Thus the above program verified and executed successfully.

EX.NO:11 **DATE:**	**SHA 1**

AIM:

To write a program to calculate the message of text using the SHA-1 algorithm.

ALGORITHM:

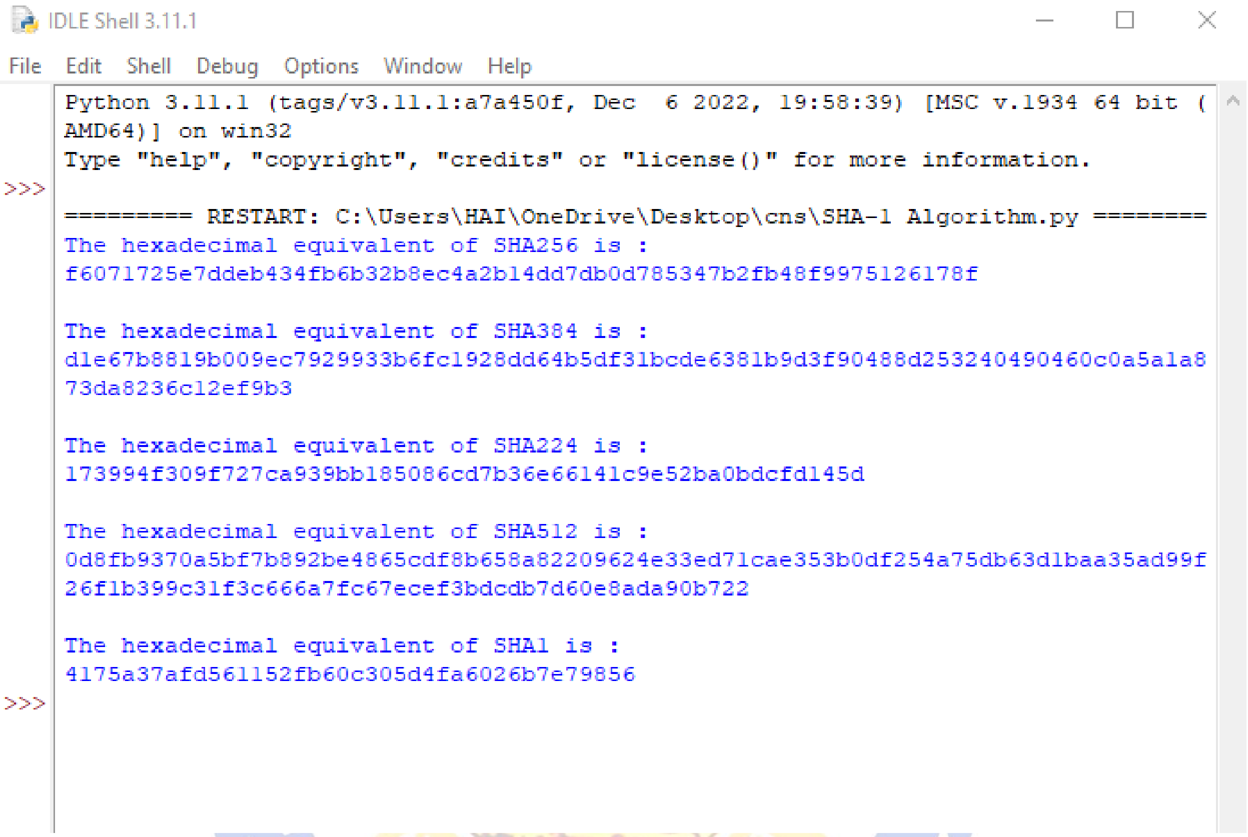
1. Start the program.
2. Declare the variables and values.
3. Using SHA-1 hash function to encrypt and decrypt the values.
4. Display the result.
5. Stop the program.

PROGRAM:

```
# Python 3 code to demonstrate
# SHA hash algorithms.

import hashlib

# initializing string
str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()
# then sending to SHA256()
result = hashlib.sha256(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA256 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
```

```
str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()
# then sending to SHA384()
result = hashlib.sha384(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA384 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()
# then sending to SHA224()
result = hashlib.sha224(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA224 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()
# then sending to SHA512()
result = hashlib.sha512(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA512 is : ")
print(result.hexdigest())

print ("\r")

# initializing string
str = "GeeksforGeeks"

# encoding GeeksforGeeks using encode()
# then sending to SHA1()
result = hashlib.sha1(str.encode())

# printing the equivalent hexadecimal value.
print("The hexadecimal equivalent of SHA1 is : ")
print(result.hexdigest())
```

OUTPUT:



```
Python 3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
========= RESTART: C:\Users\HAI\OneDrive\Desktop\cns\SHA-1 Algorithm.py ========
The hexadecimal equivalent of SHA256 is :
f6071725e7ddeb434fb6b32b8ec4a2b14dd7db0d785347b2fb48f9975126178f

The hexadecimal equivalent of SHA384 is :
d1e67b8819b009ec7929933b6fc1928dd64b5df31bcde6381b9d3f90488d253240490460c0a5a1a8
73da8236c12ef9b3

The hexadecimal equivalent of SHA224 is :
173994f309f727ca939bb185086cd7b36e66141c9e52ba0bdcfd145d

The hexadecimal equivalent of SHA512 is :
0d8fb9370a5bf7b892be4865cdf8b658a82209624e33ed71cae353b0df254a75db63d1baa35ad99f
26f1b399c31f3c666a7fc67ecef3bdcdb7d60e8ada90b722

The hexadecimal equivalent of SHA1 is :
4175a37afd561152fb60c305d4fa6026b7e79856
>>> 
```

RESULT:

Thus the above program is verified and executed successfully.

EX.NO:12 **DATE:**	**DIGITAL SIGNATURE SCHEME**

AIM:

To write a program to implement the Signature Scheme-Digital Signature Standard.

ALGORITHM:

1. Start the program.
2. Declare the variables and values.
3. Using cryptography package to generate the digital signature.
4. Using key pair () to generate the key pairs of the data.
5. Display the result
6. Stop the program.

PROGRAM:

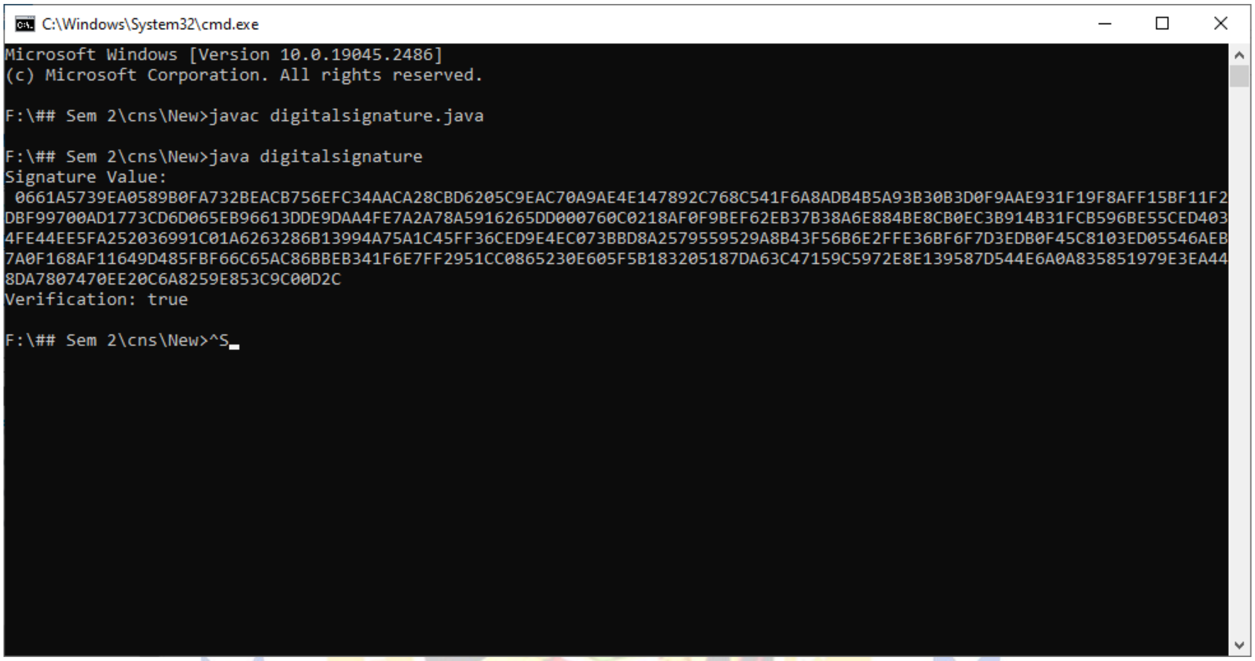
```
// Java implementation for Generating
// and verifying the digital signature
// Imports
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.SecureRandom;
import java.security.Signature;
import java.util.Scanner;
import javax.xml.bind.DatatypeConverter;

public class dss {

        // Signing Algorithm
        private static final String
            SIGNING_ALGORITHM
            = "SHA256withRSA";
        private static final String RSA = "RSA";
        private static Scanner sc;
```

```
// Function to implement Digital signature
// using SHA256 and RSA algorithm
// by passing private key.
public static byte[] Create_Digital_Signature(
byte[] input,
PrivateKey Key)
throws Exception
{
Signature signature
= Signature.getInstance(
SIGNING_ALGORITHM);
signature.initSign(Key);
signature.update(input);
return signature.sign();
}

// Generating the asymmetric key pair
// using SecureRandom class
// functions and RSA algorithm.
```

```
public static KeyPair Generate_RSA_KeyPair() throws Exception
{
    SecureRandom secureRandom
        = new SecureRandom();
    KeyPairGenerator keyPairGenerator
        = KeyPairGenerator.getInstance(RSA);
    keyPairGenerator.initialize(
            2048, secureRandom);
    return keyPairGenerator.generateKeyPair();
}

// Function for Verification of the
// digital signature by using the public key
public static boolean
Verify_Digital_Signature(
    byte[] input,
    byte[] signatureToVerify,
    PublicKey key)
    throws Exception
{
```

```
            Signature signature
                =
Signature.getInstance(SIGNING_ALGORITHM);
            signature.initVerify(key);
            signature.update(input);
            return signature.verify(signatureToVerify);
        }

        // Driver Code
        public static void main(String args[])throws Exception
        {

            String input
                = "GEEKSFORGEEKS IS A"
                + " COMPUTER SCIENCE PORTAL";
            KeyPair keyPair
                = Generate_RSA_KeyPair();

            // Function Call
            byte[] signature
                = Create_Digital_Signature(input.getBytes(),
                    keyPair.getPrivate());

            System.out.println(
                "Signature Value:\n "
                + DatatypeConverter
                    .printHexBinary(signature));

            System.out.println(
                "Verification: "
                + Verify_Digital_Signature(
                    input.getBytes(),
                    signature, keyPair.getPublic()));
        }
}
```

OUTPUT:



```
Microsoft Windows [Version 10.0.19045.2486]
(c) Microsoft Corporation. All rights reserved.

F:\## Sem 2\cns\New>javac digitalsignature.java

F:\## Sem 2\cns\New>java digitalsignature
Signature Value:
 0661A5739EA0589B0FA732BEACB756EFC34AACA28CBD6205C9EAC70A9AE4E147892C768C541F6A8ADB4B5A93B30B3D0F9AAE931F19F8AFF15BF11F2
DBF99700AD1773CD6D065EB96613DDE9DAA4FE7A2A78A5916265DD000760C0218AF0F9BEF62EB37B38A6E884BE8CB0EC3B914B31FCB596BE55CED403
4FE44EE5FA252036991C01A6263286B13994A75A1C45FF36CED9E4EC073BBD8A2579559529A8B43F56B6E2FFE36BF6F7D3EDB0F45C8103ED05546AEB
7A0F168AF11649D485FBF66C65AC86BBEB341F6E7FF2951CC0865230E605F5B183205187DA63C47159C5972E8E139587D544E6A0A835851979E3EA44
8DA7807470EE20C6A8259E853C9C00D2C
Verification: true

F:\## Sem 2\cns\New>^S
```

RESULT:

Thus the above program is verified and executed successfully.