Deccan Education Society's

Kirti M. Doongursee College of Arts, Science and Commerce

[NAAC Accredited: "A Grade"]

M.Sc. [Computer Science]

Practical Journal

PAPER: PSCSP302

Roll Number [______]

Department of Computer Science and Information Technology

Department of Computer Science and Information Technology

Deccan Education Society’s

Kirti M. Doongursee College of Arts, Science and Commerce

[NAAC Accredited: “A Grade”]

CERTIFICATE

This is to certify that Mr./Mrs. ______________________________________ of M.Sc. (Computer Science) with Roll No._________ has completed ______ Practicals of Paper **PSCSP302** under my supervision in this College during the year 2022-2023.

Dr. Neha Ansari	**Dr. Apurva Yadav**
Lecturer-In-Charge	**H.O.D.**
	Dept of CS & IT
Date:	Date:
Examined by:	**Remarks:**
Date:	_______________

Index

Aim: 1.A. Program to implement password salting and hashing to create secure password using bcrypt library.

In [1]:

```
import bcrypt
```

In [2]:

```
password = input('Enter Password: ')
pw = bytes(password, 'UTF-8')
pw
```

```
Enter Password: Practical1A
```

Out[2]:

```
b'Practical1A'
```

In [3]:

```
salt = bcrypt.gensalt()
```

In [4]:

```
hashed_pw = bcrypt.hashpw(pw, salt)
```

In [5]:

```
print('Password is ', pw, 'its salted hash is ', hashed_pw)
```

```
Password is  b'Practical1A' its salted hash is  b'$2b$12$53EGK/rKLNUAF7kFuNC92eKWZ
gx8J./mUNgLAayJq8KHjKUmyWliO'
```

Aim: 1.B. Program to implement password salting and hashing to create secure password using hashlib library.

In [1]:

```
import hashlib
```

In [2]:

```
password = input('Enter Password: ')
password
```

```
Enter Password: Practical1B
```

Out[2]:

```
'Practical1B'
```

In [3]:

```
salt = input('Enter Salt value: ')
salt
```

```
Enter Salt value: Kirti
```

Out[3]:

```
'Kirti'
```

In [4]:

```
salted_pw = password + salt
salted_pw
```

Out[4]:

```
'Practical1BKirti'
```

In [5]:

```
hashed_pw = hashlib.md5(salted_pw.encode())
hashed_pw
```

Out[5]:

```
<md5 _hashlib.HASH object @ 0x00000180F5B6CA10>
```

In [6]:

```
print(hashed_pw.hexdigest())
```

```
09c58187d3b30eeb2075f8698c1ad003
```

Aim: 2.A. Implementing substitution cipher algorithm to create cipher text.

In [1]:

```
import string
```

In [2]:

```
all_letters = string.ascii_letters
all_letters
```

Out[2]:

```
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [3]:

```
dict = {}
```

In [4]:

```
key = 3
```

In [5]:

```
for i in range(len(all_letters)):
    dict[all_letters[i]] = all_letters[(i + key) % len(all_letters)]
```

In [6]:

```
dict
```

Out[6]:

```
{'a': 'd',
 'b': 'e',
 'c': 'f',
 'd': 'g',
 'e': 'h',
 'f': 'i',
 'g': 'j',
 'h': 'k',
 'i': 'l',
 'j': 'm',
 'k': 'n',
 'l': 'o',
 'm': 'p',
 'n': 'q',
 'o': 'r',
 'p': 's',
 'q': 't',
 'r': 'u',
 's': 'v',
 't': 'w',
 'u': 'x',
 'v': 'y',
 'w': 'z',
 'x': 'A',
 'y': 'B',
 'z': 'C',
 'A': 'D',
 'B': 'E',
 'C': 'F',
 'D': 'G',
 'E': 'H',
 'F': 'I',
 'G': 'J',
 'H': 'K',
 'I': 'L',
 'J': 'M',
 'K': 'N',
 'L': 'O',
 'M': 'P',
 'N': 'Q',
 'O': 'R',
 'P': 'S',
 'Q': 'T',
 'R': 'U',
 'S': 'V',
 'T': 'W',
 'U': 'X',
 'V': 'Y',
 'W': 'Z',
 'X': 'a',
 'Y': 'b',
 'Z': 'c'}
```

In [7]:

```
plain_text = input('Enter Text: ')
plain_text
```

```
Enter Text: Practical 2A
```

Out[7]:

```
'Practical 2A'
```

In [8]:

```
cipher_text = []
```

In [9]:

```
for char in plain_text:
    if char in all_letters:
        temp = dict[char]
        cipher_text.append(temp)
    else:
        temp = char
        cipher_text.append(temp)
```

In [10]:

```
cipher_text = "".join(cipher_text)
cipher_text
```

Out[10]:

```
'Sudfwlfdo 2D'
```

Aim: 2.B. Implementing vigener cipher algorithm to create cipher text.

In [1]:

```
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))
```

In [2]:

```
answer = generateKey('CRYPTOGRAPHY', 'KIRTI')
answer
```

```
Out[2]: 'KIRTIKIRTIKI'
```

In [3]:

```
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))
```

In [4]:

```
string = input('Enter Text: ')
string
```

```
        Enter Text: Practical 2B
Out[4]: 'Practical 2B'
```

In [5]:

```
keyword = input('Enter Key: ')
keyword
```

```
        Enter Key: Kirti
Out[5]: 'Kirti'
```

In [6]:

```
key = generateKey(string, keyword)
key
```

```
Out[6]: 'KirtiKirtiKi'
```

In [7]:

```
cipher = cipherText(string, key)
cipher
```

```
Out[7]: 'ZLDHNYWDQHVP'
```

```
In [8]: print('Plain Text: ', string)
```

```
Plain Text:  Practical 2B
```

```
In [9]: print('Cipher Text: ', cipher)
```

```
Cipher Text:  ZLDHNYWDQHVP
```

Aim: 3.A. Decrypt analysis of cipher text generated using substitution cipher.

In [1]:

```
import string
```

In [2]:

```
all_letters = string.ascii_letters
all_letters
```

Out[2]:

```
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

In [3]:

```
dict = {}
```

In [4]:

```
key = 3
```

In [5]:

```
for i in range(len(all_letters)):
    dict[all_letters[i]] = all_letters[(i - key) % len(all_letters)]
```

In [6]:

```
dict
```

Out[6]:

```
{'a': 'X',
 'b': 'Y',
 'c': 'Z',
 'd': 'a',
 'e': 'b',
 'f': 'c',
 'g': 'd',
 'h': 'e',
 'i': 'f',
 'j': 'g',
 'k': 'h',
 'l': 'i',
 'm': 'j',
 'n': 'k',
 'o': 'l',
 'p': 'm',
 'q': 'n',
 'r': 'o',
 's': 'p',
 't': 'q',
 'u': 'r',
 'v': 's',
 'w': 't',
 'x': 'u',
 'y': 'v',
 'z': 'w',
 'A': 'x',
 'B': 'y',
 'C': 'z',
 'D': 'A',
 'E': 'B',
 'F': 'C',
 'G': 'D',
 'H': 'E',
 'I': 'F',
 'J': 'G',
 'K': 'H',
 'L': 'I',
 'M': 'J',
 'N': 'K',
 'O': 'L',
 'P': 'M',
 'Q': 'N',
 'R': 'O',
 'S': 'P',
 'T': 'Q',
 'U': 'R',
 'V': 'S',
 'W': 'T',
 'X': 'U',
 'Y': 'V',
 'Z': 'W'}
```

In [7]:

```
decrypt_text = []
```

In [8]:

```
cipher_text = 'Sudfwlfdo 3D'
```

In [9]:

```
for char in cipher_text:
    if char in all_letters:
        temp = dict[char]
        decrypt_text.append(temp)
    else:
        temp = char
        decrypt_text.append(temp)
```

In [10]:

```
decrypted_text = "".join(decrypt_text)
decrypted_text
```

Out[10]:

```
'Practical 3A'
```

Aim: 3.B. Decrypt analysis of cipher text generated using vigener cipher.

In [1]:

```
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))
```

In [2]:

```
answer = generateKey('GEEKSFORGEEKS', 'AYUSH')
answer
```

Out[2]:

```
'AYUSHAYUSHAYU'
```

In [3]:

```
string = 'GEEKSFORGEEKS'
string
```

Out[3]:

```
'GEEKSFORGEEKS'
```

In [4]:

```
key = 'AYUSH'
key
```

Out[4]:

```
'AYUSH'
```

In [5]:

```
keyword = generateKey('GCYCZFMLYLEIM', key)
keyword
```

Out[5]:

```
'AYUSHAYUSHAYU'
```

In [6]:

```
cipher_text = 'GCYCZFMLYLEIM'
cipher_text
```

Out[6]:

```
'GCYCZFMLYLEIM'
```

In [7]:

```
def decrypt_cipher(cipher_text, key):
    decrypted_text = []
    for i in range(len(cipher_text)):
        x = (ord(cipher_text[i]) - ord(key[i]) + 26) % 26
        x += ord('A')
        decrypted_text.append(chr(x))
    return("".join(decrypted_text))
```

In [8]:

```
decrypt_text = decrypt_cipher(cipher_text, keyword)
decrypt_text
```

Out[8]:

```
'GEEKSFORGEEKS'
```