CNS LAB 2

SAHIL BONDRE: U18CO021

Program-1 Implement columnar transposition cipher.
a. Write a program for Encryption using key
b. Write a program for decryption using key

Program-2 Implement rail fence transposition cipher
a. Write a program for Encryption using key (depth)
b. Write a program for decryption using key (depth)

Columnar Cipher:

```
import math
from tabulate import tabulate


def recursive_read(allowed_input, message=""):
    # Recursively reads user input until input is not in allowed_input
    while True:
        user_input = input(message)
        if user_input in allowed_input:
            return user_input


def recursive_read_int(message=""):
    # Recursively reads user input until input is not in allowd_input
    while True:
        user_input = input(message)
        try:
            value = int(user_input)
            return value
```

```
        except:
            pass

def file_to_str(filename):
    try:
        with open(filename, 'r') as file:
            return file.read()
    except:
        print("Error: File not found!")
        exit(1)

def perform_encryption():
    filename = input("Enter file to be encrypted: ")
    message = file_to_str(filename)
    key = recursive_read_int("Enter key value: ")
    result = ""

    table = []

    size = len(message)
    row_count = math.ceil(size / key)
    column_count = key
    extra_bits = row_count * column_count - size

    for _ in range(extra_bits):
      message += " "
```

```
    k = 0

    for _ in range(row_count):
        table.append([])

    for i in range(row_count):
        for _ in range(column_count):
            table[i].append(message[k])
            k += 1

    print(tabulate(table, tablefmt="pretty"))
    for i in range(key):
        for row in table:
            result += row[i]
    print(f"Final string:\n{result}")

def perform_decryption():
    filename = input("Enter file to be decrypted: ")
    message = file_to_str(filename)
    key = recursive_read_int("Enter key value: ")
    result = ""

    table = []

    size = len(message)
```

```
    row_count = math.ceil(size / key)
    column_count = key
    extra_bits = row_count * column_count - size

    for _ in range(extra_bits):
      message += " "

    k = 0

    for _ in range(row_count):
      table.append([])

    for _ in range(column_count):
      for j in range(row_count):
        table[j].append(message[k])
        k += 1

    print(tabulate(table, tablefmt="pretty"))
    for row in table:
      for i in range(key):
        result += row[i]
    print(f"Final string:\n{result}")

is_encrypt = recursive_read(
    ["e", "d"], "Enter 'e' for encryption or 'd' for decryption: ") == "e"
```

```
if is_encrypt:

    perform_encryption()

else:

    perform_decryption()
```

message:

```
Sahil Bondre
Hello World!
1234657980
```

after encryption:

```
Sln
lo!37a dHor
49hBre l168ioelWd250
```

```
Enter 'e' for encryption or 'd' for decryption: e
Enter file to be encrypted: msg.txt
Enter key value: 4
+---+---+---+---+
| S | a | h | i |
| l |   | B | o |
| n | d | r | e |
|   | H | e | l |
| l | o |   | W |
| o | r | l | d |
| ! |   | 1 | 2 |
| 3 | 4 | 6 | 5 |
| 7 | 9 | 8 | 0 |
+---+---+---+---+
Final string:
Sln
lo!37a dHor
49hBre l168ioelWd250
PS F:\code\github.com\godcrampy\college-notes\cns\lab-02> python .\columnar.py
Enter 'e' for encryption or 'd' for decryption: d
Enter file to be decrypted: col.txt
Enter key value: 4
+---+---+---+---+
| S | a | h | i |
| l |   | B | o |
| n | d | r | e |
|   | H | e | l |
| l | o |   | W |
| o | r | l | d |
| ! |   | 1 | 2 |
| 3 | 4 | 6 | 5 |
| 7 | 9 | 8 | 0 |
+---+---+---+---+
Final string:
Sahil Bondre
Hello World!
1234657980
```

Railfence Cipher:

```python
import math

from tabulate import tabulate



def recursive_read(allowed_input, message=""):

    # Recursively reads user input until input is not in allowed_input

    while True:

        user_input = input(message)

        if user_input in allowed_input:
```

```
            return user_input


def recursive_read_int(message=""):
    # Recursively reads user input until input is not in allowd_input
    while True:
        user_input = input(message)
        try:
            value = int(user_input)
            return value
        except:
            pass



def file_to_str(filename):
    try:
        with open(filename, 'r') as file:
            return file.read()
    except:
        print("Error: File not found!")
        exit(1)


def perform_encryption():
    filename = input("Enter file to be encrypted: ")
    message = file_to_str(filename)
    key = recursive_read_int("Enter key value: ")
    result = ""
```

```
table = []

size = len(message)
row_count = key
column_count = math.ceil(size / key)
extra_bits = row_count * column_count - size

for _ in range(extra_bits):
  message += " "

k = 0

for _ in range(row_count):
  table.append([])

for _ in range(column_count):
  for j in range(row_count):
    table[j].append(message[k])
    k += 1

print(tabulate(table, tablefmt="pretty"))
for row in table:
  for i in range(column_count):
    result += row[i]
print(f"Final string:\n{result}")
```

```
def perform_decryption():
    filename = input("Enter file to be decrypted: ")
    message = file_to_str(filename)
    key = recursive_read_int("Enter key value: ")
    result = ""

    table = []

    size = len(message)
    row_count = key
    column_count = math.ceil(size / key)
    extra_bits = row_count * column_count - size

    for _ in range(extra_bits):
      message += " "

    k = 0

    for _ in range(row_count):
      table.append([])

    for i in range(row_count):
      for _ in range(column_count):
        table[i].append(message[k])
```

```
        k += 1


    print(tabulate(table, tablefmt="pretty"))
    for i in range(column_count):
      for row in table:
        result += row[i]
    print(f"Final string:\n{result}")


is_encrypt = recursive_read(
    ["e", "d"], "Enter 'e' for encryption or 'd' for decryption: ") == "e"


if is_encrypt:
    perform_encryption()
else:
    perform_decryption()
```

message:

```
Sahil Bondre
Hello World!
1234657980
```

after encryption:

```
SiBd
l r!269alorHlWl
358h neeood1470
```

```
PS F:\code\github.com\godcrampy\college-notes\cns\lab-02> python .\railfence.py
Enter 'e' for encryption or 'd' for decryption: e
Enter file to be encrypted: msg.txt
Enter key value: 3
+---+---+---+---+---+---+---+---+---+---+---+---+
| S | i | B | d |   | l |   | r | ! | 2 | 6 | 9 |
| a | l | o | r | H | l | W | l |   | 3 | 5 | 8 |
| h |   | n | e | e | o | o | d | 1 | 4 | 7 | 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+
Final string:
SiBd
l r!269alorHlWl
358h neeood1470
PS F:\code\github.com\godcrampy\college-notes\cns\lab-02> python .\railfence.py
Enter 'e' for encryption or 'd' for decryption: d
Enter file to be decrypted: rail.txt
Enter key value: 3
+---+---+---+---+---+---+---+---+---+---+---+---+
| S | i | B | d |   | l |   | r | ! | 2 | 6 | 9 |
| a | l | o | r | H | l | W | l |   | 3 | 5 | 8 |
| h |   | n | e | e | o | o | d | 1 | 4 | 7 | 0 |
+---+---+---+---+---+---+---+---+---+---+---+---+
Final string:
Sahil Bondre
Hello World!
1234657980
PS F:\code\github.com\godcrampy\college-notes\cns\lab-02>
```