Roll. No. A016	Name: Varun Khadayate
Class B.Tech CsBs	Batch: 1
Date of Experiment: 20-08-2021	Subject: Cryptology

Theory

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message “meet me after the toga party” with a rail fence of depth 2, we write the following:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

```
Key:          4 3 1 2 5 6 7
Plaintext:    a t t a c k p
              o s t p o n ed u n
              t i l tw o a m x y
              z
   Ciphertext:   TTNaaPTMTSUOaODWCOIxKNLyPETz
```

Thus, in this example, the key is 4312567. To encrypt, start with the column that is labeled 1, in this case column 3. Write down all the letters in that column. Proceed to column 4, which is labeled 2, then column 2, then column 1, then columns 5, 6, and 7.

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is straightforward and involves laying out the cipher- text in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm,

```
        Key:       4 3 1 2 5 6 7
        Input:     t t n a a p t
                   m t s u o a o
                   d w c o i x k
                   n l y p e t z
      Output:    NSCyaUOPTTWLTMDNaOIEPaxTTOKz
```

To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

```
01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28
```

After the first transposition, we have

```
03 10 17 24 04 11 18 25 02 09 16 23 01 08
15 22 05 12 19 26 06 13 20 27 07 14 21 28
```

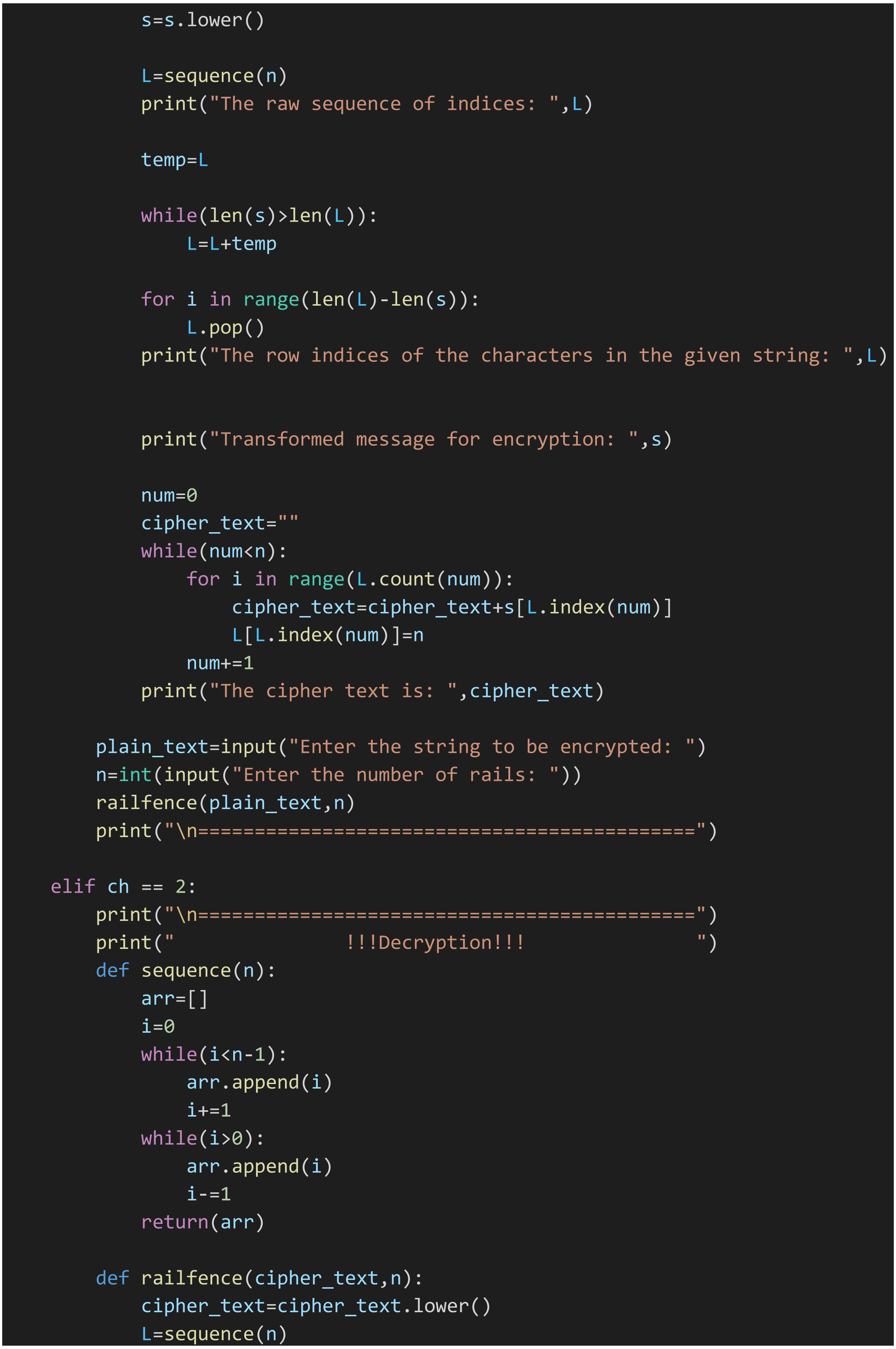
which has a somewhat regular structure. But after the second transposition, we have

```
17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28
```

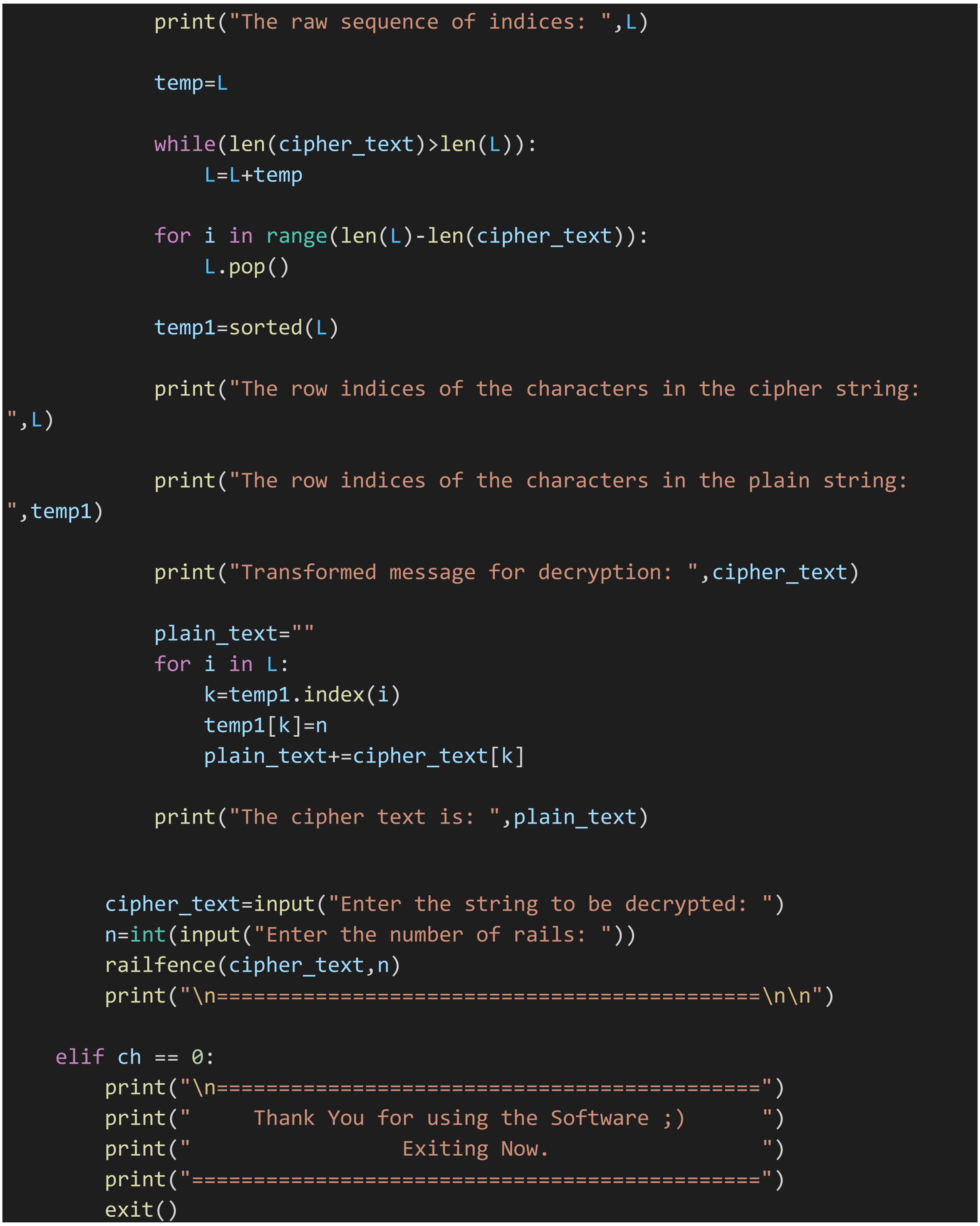
This is a much less structured permutation and is much more difficult to cryptanalyze.

Code

```
import string
import numpy as np

while True:
    ch = int(input('Welcome to Rail Fence Cipher Encryption and Decryption Program Made by Varun Khadayate..\n [*] Press 1 for Encryption \n [*] Press 2 for Decryption \n [*] Press 0 to exit..\n \nYour Choice:: '))

    if ch == 1:
        print("============================================")
        print("             !!!!Encryption!!!!             ")
        def sequence(n):
            arr=[]
            i=0
            while(i<n-1):
                arr.append(i)
                i+=1
            while(i>0):
                arr.append(i)
                i-=1
            return(arr)

        def railfence(s,n):
```

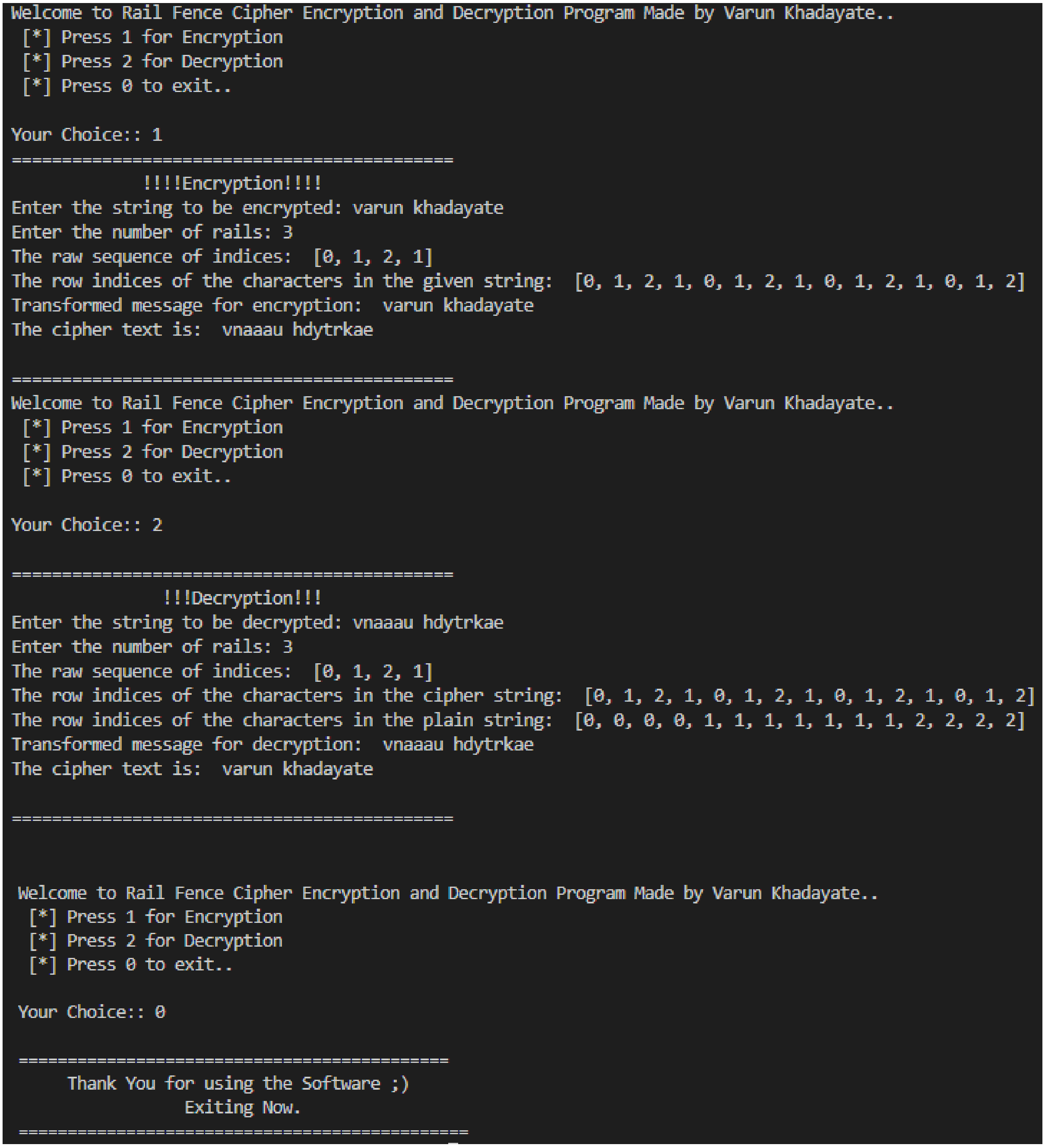


```
            s=s.lower()

            L=sequence(n)
            print("The raw sequence of indices: ",L)

            temp=L

            while(len(s)>len(L)):
                L=L+temp

            for i in range(len(L)-len(s)):
                L.pop()
            print("The row indices of the characters in the given string: ",L)


            print("Transformed message for encryption: ",s)

            num=0
            cipher_text=""
            while(num<n):
                for i in range(L.count(num)):
                    cipher_text=cipher_text+s[L.index(num)]
                    L[L.index(num)]=n
                num+=1
            print("The cipher text is: ",cipher_text)

        plain_text=input("Enter the string to be encrypted: ")
        n=int(input("Enter the number of rails: "))
        railfence(plain_text,n)
        print("\n============================================")

    elif ch == 2:
        print("\n============================================")
        print("               !!!Decryption!!!               ")
        def sequence(n):
            arr=[]
            i=0
            while(i<n-1):
                arr.append(i)
                i+=1
            while(i>0):
                arr.append(i)
                i-=1
            return(arr)

        def railfence(cipher_text,n):
            cipher_text=cipher_text.lower()
            L=sequence(n)
```



```
            print("The raw sequence of indices: ",L)

            temp=L

            while(len(cipher_text)>len(L)):
                L=L+temp

            for i in range(len(L)-len(cipher_text)):
                L.pop()

            temp1=sorted(L)

            print("The row indices of the characters in the cipher string:
",L)

            print("The row indices of the characters in the plain string:
",temp1)

            print("Transformed message for decryption: ",cipher_text)

            plain_text=""
            for i in L:
                k=temp1.index(i)
                temp1[k]=n
                plain_text+=cipher_text[k]

            print("The cipher text is: ",plain_text)


        cipher_text=input("Enter the string to be decrypted: ")
        n=int(input("Enter the number of rails: "))
        railfence(cipher_text,n)
        print("\n============================================\n\n")

    elif ch == 0:
        print("\n============================================")
        print("     Thank You for using the Software ;)      ")
        print("                 Exiting Now.                 ")
        print("==============================================")
        exit()
```

Output



```
Welcome to Rail Fence Cipher Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 1
============================================
             !!!!Encryption!!!!
Enter the string to be encrypted: varun khadayate
Enter the number of rails: 3
The raw sequence of indices:  [0, 1, 2, 1]
The row indices of the characters in the given string:  [0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2]
Transformed message for encryption:  varun khadayate
The cipher text is:  vnaaau hdytrkae

============================================
Welcome to Rail Fence Cipher Encryption and Decryption Program Made by Varun Khadayate..
 [*] Press 1 for Encryption
 [*] Press 2 for Decryption
 [*] Press 0 to exit..

Your Choice:: 2

============================================
               !!!Decryption!!!
Enter the string to be decrypted: vnaaau hdytrkae
Enter the number of rails: 3
The raw sequence of indices:  [0, 1, 2, 1]
The row indices of the characters in the cipher string:  [0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2, 1, 0, 1, 2]
The row indices of the characters in the plain string:  [0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2]
Transformed message for decryption:  vnaaau hdytrkae
The cipher text is:  varun khadayate

============================================



 Welcome to Rail Fence Cipher Encryption and Decryption Program Made by Varun Khadayate..
  [*] Press 1 for Encryption
  [*] Press 2 for Decryption
  [*] Press 0 to exit..

 Your Choice:: 0

 ===========================================
      Thank You for using the Software ;)
                 Exiting Now.
 =============================================
```