KADI SARVA VISHWAVIDYALAYA
LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

Department of
Computer Engineering and Information Technology

Subject: Data Compression (IT603-N)

Laboratory Manual

Prepared By
Patel Joy
19BEIT30033
6th IT

LDRP INSTITUTE OF TECHNOLOGY AND RESEARCH GANDHINAGAR

DEPARTMENT OF COMPUTER ENGINEERING
&
INFORMATION TECHNOLOGY

CERTIFICATE

Mr./Miss____

of 6th IT Enrolment No.__________________________________

Exam No, ____________________has satisfactorily completed his/her term work in ***Data Compression (IT603-N)*** for the term ending in **Dec-May-2022**.

Date: _______________

Dr. Mehul Barot

Subject Coordinator **HOD-IT**

INDEX

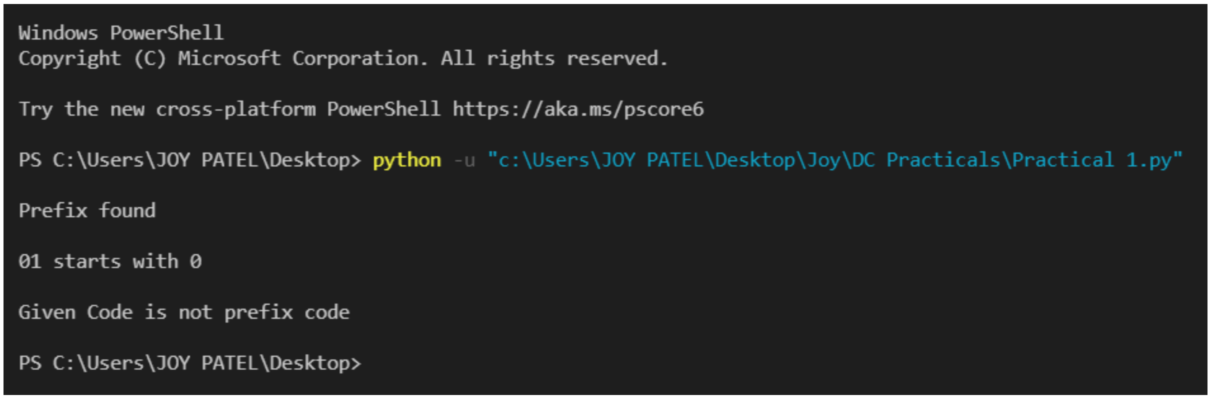
Practical 1

Aim: Write a program to check whether the given code is Prefix or not

Program:

```
a=["0","1","01","11"]
flag=1
for x in range(len(a)):
    for y in range(x+1,len(a)):
        if a[y].startswith(a[x]):
            flag=0
            print("\nPrefix found\n")
            print(a[y] + " starts with " + a[x])
            break
    if(not flag):
        print("\nGiven Code is not prefix code\n")
        break
else:
    print("\nWe  didn't find a  prefix. So it is a prefix code\n")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\JOY PATEL\Desktop> python -u "c:\Users\JOY PATEL\Desktop\Joy\DC Practicals\Practical 1.py"

Prefix found

01 starts with 0

Given Code is not prefix code

PS C:\Users\JOY PATEL\Desktop>
```

Conclusion: We have learnt how to implement a program to check whether a given code is prefix or not using python

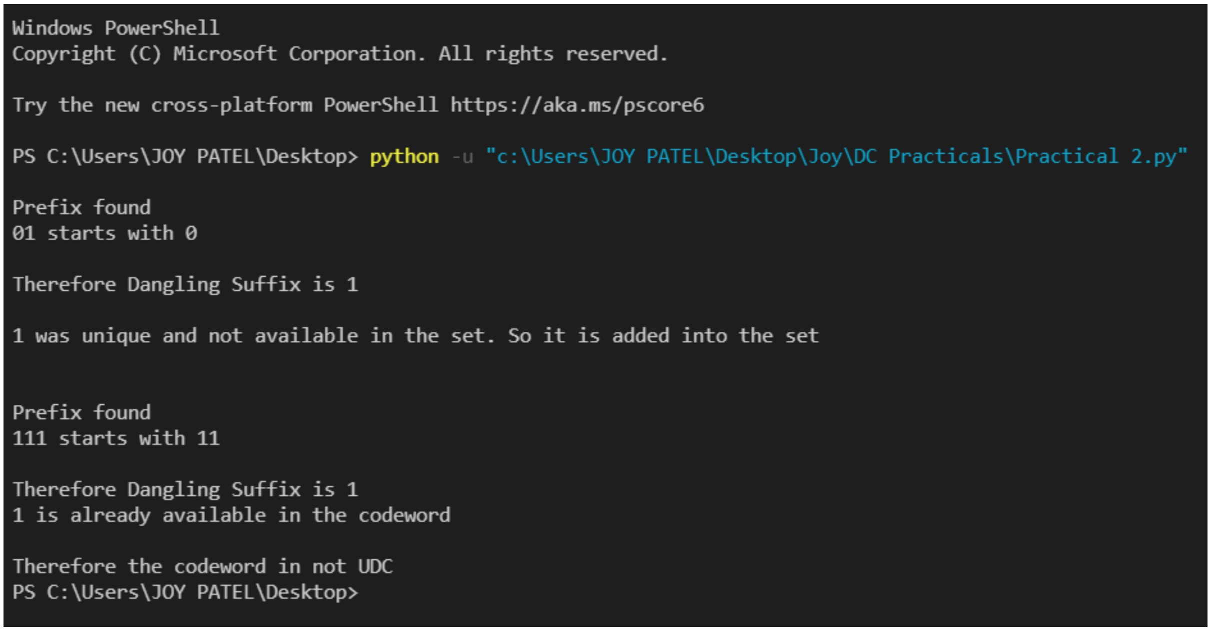
Practical 2

Aim: Write a program to check whether the set of given codes is Uniquely Decodable or not

Program:

```
a=["0","01","11","111"]

flag1=1

flag2=1


for x in range(len(a)):

   for y in range(x+1,len(a)):

      if flag2:

         if a[y].startswith(a[x]):

            flag1=0

            print("\nPrefix found")

            print(a[y] + " starts with " + a[x])

            dang_suffix=a[y].removeprefix(a[x])

            print("\nTherefore Dangling Suffix is " + dang_suffix)

            if dang_suffix in a:

               flag2=0

               print(dang_suffix + " is already available in the codeword")

               print("\nTherefore the codeword in not UDC")

            else:

               print("\n" +dang_suffix + " was unique and not available in the set. So it is added into the set\n")

               a.append(dang_suffix)


if flag1:

   print("\nNo prefix was found. So this is a UDC\n")

if flag2:

   print("\nGiven code is a UDC\n")
```

Output:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\JOY PATEL\Desktop> python -u "c:\Users\JOY PATEL\Desktop\Joy\DC Practicals\Practical 2.py"

Prefix found
01 starts with 0

Therefore Dangling Suffix is 1

1 was unique and not available in the set. So it is added into the set


Prefix found
111 starts with 11

Therefore Dangling Suffix is 1
1 is already available in the codeword

Therefore the codeword in not UDC
PS C:\Users\JOY PATEL\Desktop>
```

Conclusion: We have learnt how to implement a program to check whether a given set of codes is Uniquely Decodable or not using python

Practical 3

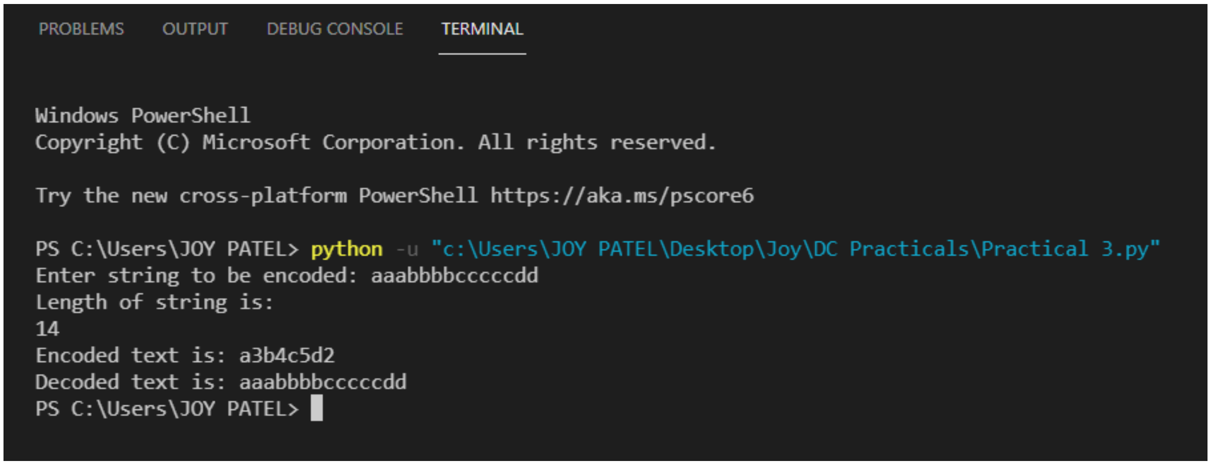
Aim: Write a program to compress and decompress the given input string. (Using Run-length Coding)

Program:

```
str1=input("Enter string to be encoded: ")
cipherlist=[]
i=0
length=len(str1)
print(“Length of string is:  ”)
print(length)
while i<length:
   symbol=str1[i]
   count=1
   j=i
   while j<length-1 and (str1[j+1]==str1[j]):
      count+=1
      j=j+1
   cipherlist.append(symbol)
   cipherlist.append(str(count))
   i=j+1

encoded_text=""
i=0
while i<len(cipherlist) :
   encoded_text+=cipherlist[i]
   i=i+1

print("Encoded text is: " + encoded_text)

decoded_text=""
i=0
```

```
while i<len(cipherlist):
   j=cipherlist[i+1]
   n=int(j)
   while n>0:
      decoded_text+=cipherlist[i]
      n=n-1
   i=i+2

print("Decoded text is: " + decoded_text)
```

OUTPUT:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\JOY PATEL> python -u "c:\Users\JOY PATEL\Desktop\Joy\DC Practicals\Practical 3.py"
Enter string to be encoded: aaabbbbcccccdd
Length of string is:
14
Encoded text is: a3b4c5d2
Decoded text is: aaabbbbcccccdd
PS C:\Users\JOY PATEL> █
```

Conclusion: We have learnt how to implement a program to compress and decompress the given input string using Run-length Coding

Practical 4

Aim: Write a program to generate Shannon- Fano Code and encode file using generated code and find compression ratio, average length and redundancy

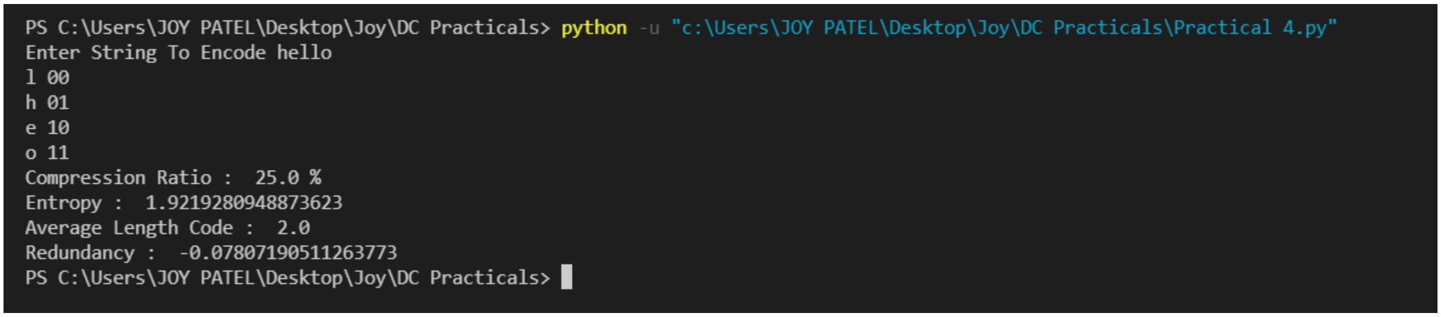
Program:

```
import math


result=[]


def split(l,p,q):
  if len(l) <=1:
    return
  else:
    b=[]
    c=[]
    d=0
    e=0

  for i in l:
    if d==e or e+i>d:
      b.append(i)
      d=d+i
    else:
      c.append(i)
      e=e+i

  if len(b)==1:
    result.append(p)
  if len(c)==1:
    result.append(q)
  split(b,p+'0',p+'1')
  split(c,q+'0',q+'1')
```

```
a=input("Enter String To Encode ")
s=[]

for i in a:
    if i not in s:
        s.append(i)

l=[]
prob=[]

for i in s:
    l.append(a.count(i))

for i in range(len(l)-1):
    c=0
    for j in range(0,len(l)-(i+1)):
        if l[j]<l[j+1]:
            b=l[j]
            z=s[j]
            l[j]=l[j+1]
            s[j]=s[j+1]
            l[j+1]=b
            s[j+1]=z
            c=1
    if(c==0):
            break

for i in l:
    prob.append(i/len(a))
```

```
p='0'

q='1'

split(l,p,q)

c=0

z=0

y=0


for i in range(len(s)):

   c=c+(len(result[i])*l[i])

   print(s[i],result[i])

   y=y+len(result[i])*prob[i]

   z=z+(prob[i]*(math.log(prob[i],2)))


print('Compression Ratio : ',(c/(len(a)*8))*100,'%')

print('Entropy : ',-1*z)

print('Average Length Code : ',y)

print('Redundancy : ',(-1*z)-y)
```

OUTPUT:



```
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> python -u "c:\Users\JOY PATEL\Desktop\Joy\DC Practicals\Practical 4.py"
Enter String To Encode hello
l 00
h 01
e 10
o 11
Compression Ratio :  25.0 %
Entropy :  1.9219280948873623
Average Length Code :  2.0
Redundancy :  -0.07807190511263773
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals>
```

Conclusion: Shanon - fanon coding is a simple method to compress the data where alphabets are converted into a numeric code so as to reduce the memory usage .

Practical 5

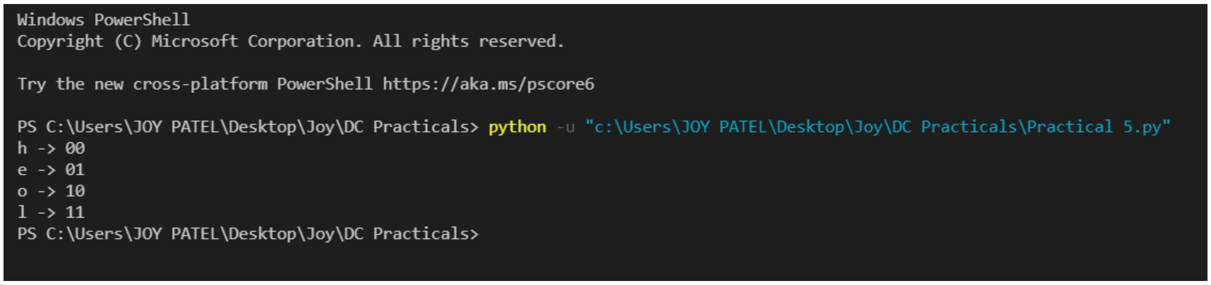
Aim: Write a program for implementing the Huffman Coding.

Program:

```
class node:
    def init (self, freq, symbol, left=None, right=None):
        self.freq = freq
        self.symbol = symbol
        self.left = left
        self.right = right
        self.huff = ''

def printNodes(node, val=''):
    newVal = val + str(node.huff)
    if(node.left):
        printNodes(node.left, newVal)

    if(node.right):
        printNodes(node.right, newVal)

    if(not node.left and not node.right):
        print(f"{node.symbol} -> {newVal}")

a=input("Enter String To Encode ")
chars=[]

for i in a:
    if i not in chars:
        chars.append(i)
```

```
freq=[]
for i in chars:
    freq.append(a.count(i))
nodes = []

for x in range(len(chars)):
    nodes.append(node(freq[x],chars[x]))

while len(nodes) > 1:
    nodes = sorted(nodes, key=lambda x: x.freq)
    left = nodes[0]
    right = nodes[1]
    left.huff = 0
    right.huff = 1
    newNode = node(left.freq+right.freq, left.symbol+right.symbol, left, right)
    nodes.remove(left)
    nodes.remove(right)
    nodes.append(newNode)
printNodes(nodes[0])
```

OUTPUT:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> python -u "c:\Users\JOY PATEL\Desktop\Joy\DC Practicals\Practical 5.py"
h -> 00
e -> 01
o -> 10
l -> 11
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals>
```

Conclusion: Huffman coding is a simple method to compress the data where alphabets are converted into a numeric code in tree format so as to reduce the memory usage .

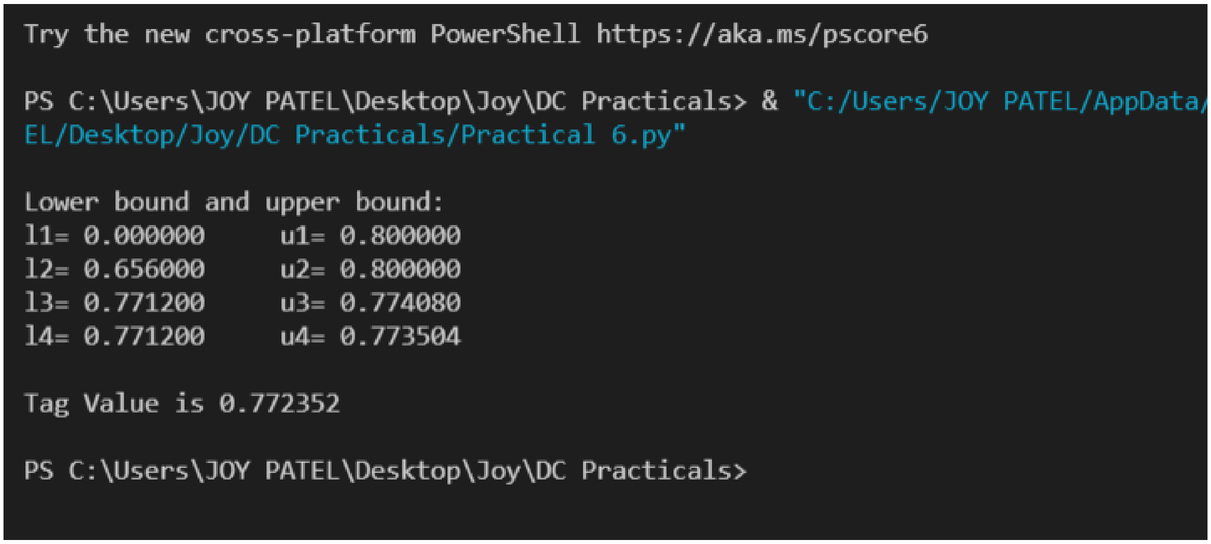
Practical 6

Aim: Write a program to implement Arithmetic Coding Compression

Program:

```
seq=[0,1,3,2,1]
prob=[0,0.8,0.02,0.18]
fx=[0]*len(prob)
for i in range(1,len(prob)):
   j=i
   while j>0:
      fx[i]= round(fx[i],2)+ round(prob[j],2)
      fx[i]=round(fx[i],3)
      j-=1

lb=[0]*(len(seq))
ub=[1]*(len(seq))

for i in range(1,len(seq)):
   lb[i]=lb[i-1]+ (ub[i-1] - lb[i-1]) * fx[(seq[i]-1)]
   ub[i]=lb[i-1]+ (ub[i-1] - lb[i-1]) * fx[(seq[i])]

print("\nLower bound and upper bound:")
for i in range(1,len(lb)):
   print("l{}= {:.6f} \t u{}= {:.6f}".format(i,lb[i],i,ub[i]))

tagValue=(lb[4]+ub[4])/2
print("\nTag Value is "+ format(tagValue)+"\n")
```

OUTPUT:



```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> & "C:/Users/JOY PATEL/AppData/
EL/Desktop/Joy/DC Practicals/Practical 6.py"

Lower bound and upper bound:
l1= 0.000000     u1= 0.800000
l2= 0.656000     u2= 0.800000
l3= 0.771200     u3= 0.774080
l4= 0.771200     u4= 0.773504

Tag Value is 0.772352

PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals>
```

Conclusion:

Arithmetic Coding generates a variable length code. It is more efficient with the source having small size of sequence’s. It is used for numerical as well as alphabetical encoding. It generates a tag value between 0 and 1 at the end of encoding using which the sequence can be decoded.

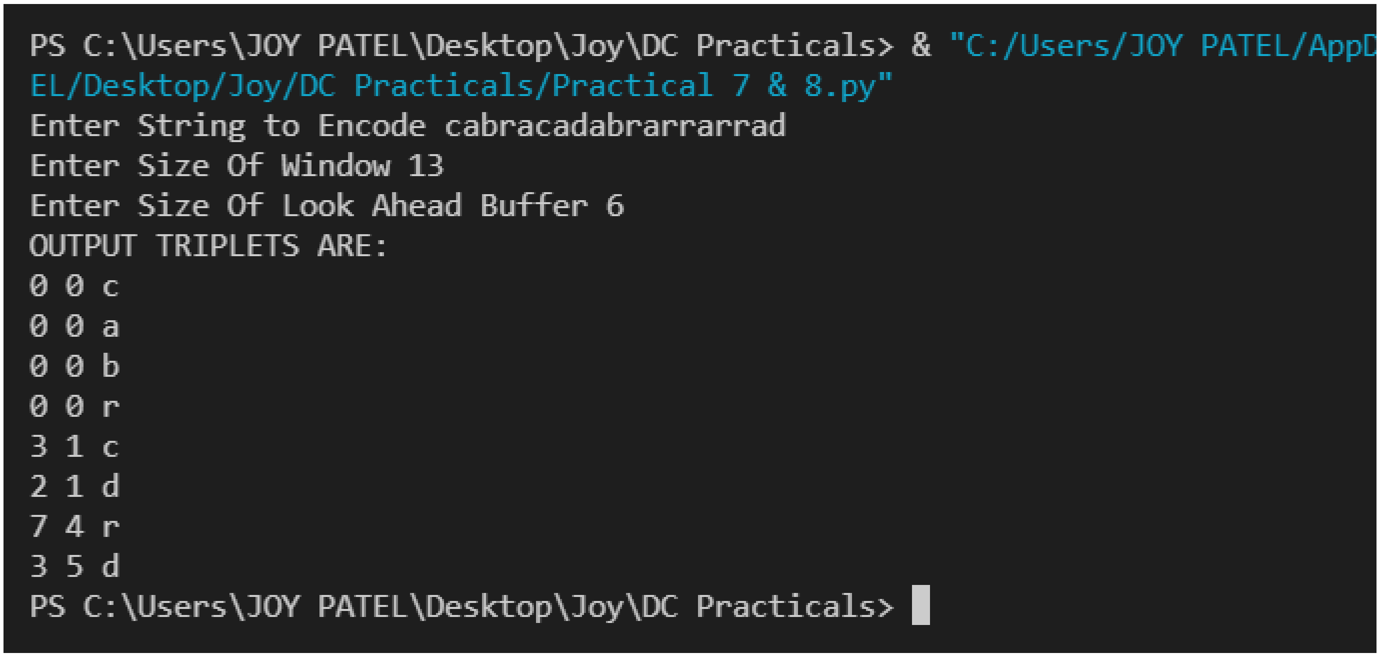
Practical 7

Aim: Write a program to implement LZ77 compression algorithm.

Program:

```
a=input('Enter String to Encode ')
b=int(input('Enter Size Of Window '))
c=int(input('Enter Size Of Look Ahead Buffer '))
l=[]
le=[]
r=[]

for i in range(len(a)):
    if i>=c:
        l.append(a[i])
    else:
        r.append(a[i])
print('OUTPUT TRIPLETS ARE: ')
while(len(r)>0):
    if r[0] in le:
        j=0
        i=0
        z=0
        y=0
        m=0
        while i<len(le):
            if r[j]==le[i]:
                j=j+1
                z=z+1
            else:
                k=j
                q=z
                for p in range(k):
                    if r[p]==r[j]:
                        j=j+1
                        z=z+1
                        if(j>=len(r)):
                            break
                    else:
                            break
                if(z>=y):
                    y=z
                    m=abs(i-q-len(le))

                j=0
            z=0
        i=i+1
    if j!=0:
        q=z
        k=j
        for p in range(k):
            if r[p]==r[j]:
                j=j+1
                z=z+1
                if(j>=len(r)):
                 break
            else:
                break
        y=z
        m=abs(i-q-len(le))
    for i in range(y+1):
        if (len(le)>=(b-c)):
            le.pop(0)
        le.append(r[0])
        r.pop(0)
        if len(l)!=0:
            r.append(l[0])
            l.pop(0)
    print(m,y,le[-1])

else:
    print(0,0,r[0])
    if len(le)>(b-c):
        le.pop(0)
    le.append(r[0])
    r.pop(0)
    if len(l)!=0:
        r.append(l[0])
        l.pop(0)
```

OUTPUT:



```
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> & "C:/Users/JOY PATEL/AppD
EL/Desktop/Joy/DC Practicals/Practical 7 & 8.py"
Enter String to Encode cabracadabrarrarrad
Enter Size Of Window 13
Enter Size Of Look Ahead Buffer 6
OUTPUT TRIPLETS ARE:
0 0 c
0 0 a
0 0 b
0 0 r
3 1 c
2 1 d
7 4 r
3 5 d
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> 
```

Conclusion: LZ77 is an algorithm in which there is a fix window where all elements are passed through and gets compressed according to the pattern repetition and final answer is obtained as triplets.

Practical 8

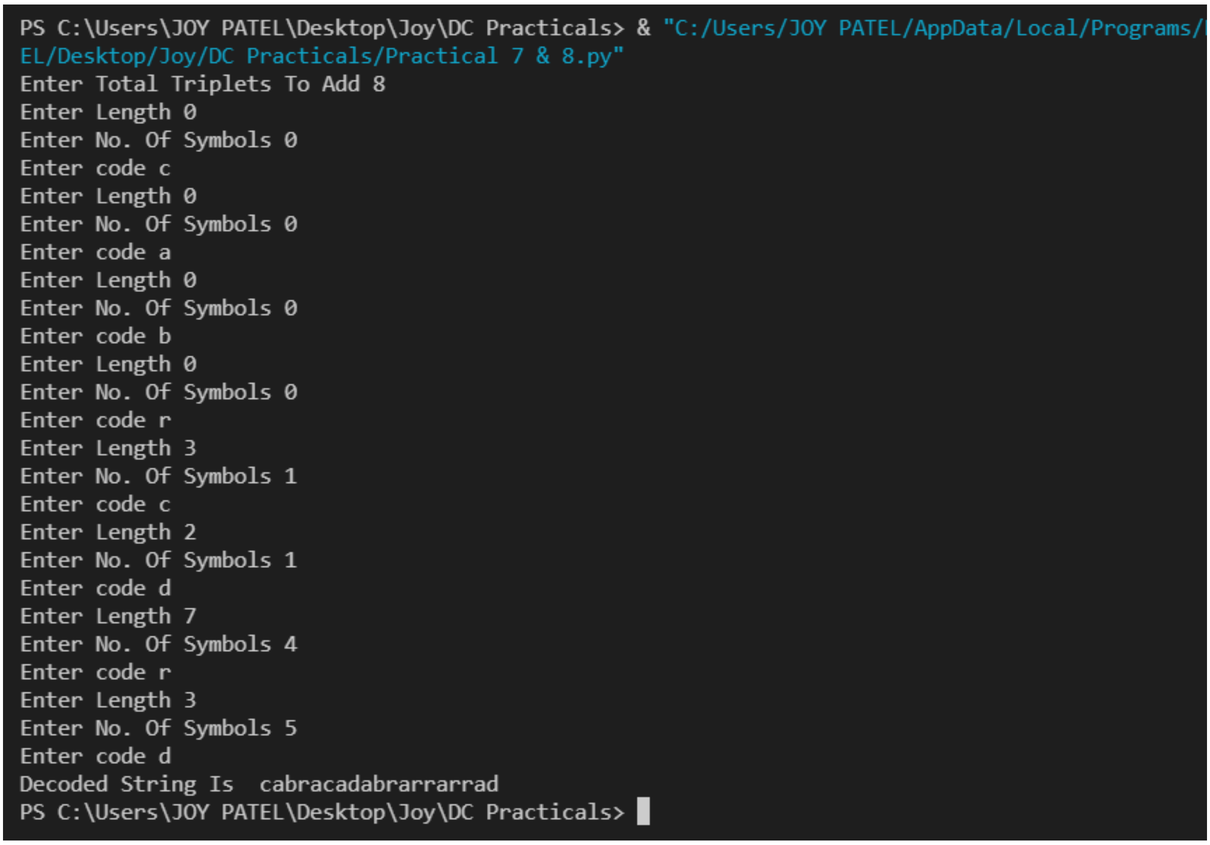
Aim: Write a program to implement LZ77 decompression algorithm.

Program:

```
a=int(input('Enter Total Triplets To Add '))
l=[]
for i in range(a):
   b=[]
   b.append(int(input('Enter Length ')))
   b.append(int(input('Enter No. Of Symbols ')))
   b.append(input('Enter code '))
   l.append(b)

f=[]
for i in l:
   b=[]
   if i[1]>i[0]:
      k=0
      j=-(i[0])
      while(k<i[1]):
         if(j>-1):
            j=-(i[0])
         e=f[j]
         b.append(e)
         j=j+1
         k=k+1
      b.append(i[2])
   else:
      for j in range(-(i[0]),(-(i[0])+i[1])):
         e=f[j]
         b.append(e)
```

```
        b.append(i[2])
    f.extend(b)
s=''
for i in f:
    s=s+i
print('Decoded String Is ',s)
```

OUTPUT:



```
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> & "C:/Users/JOY PATEL/AppData/Local/Programs/
EL/Desktop/Joy/DC Practicals/Practical 7 & 8.py"
Enter Total Triplets To Add 8
Enter Length 0
Enter No. Of Symbols 0
Enter code c
Enter Length 0
Enter No. Of Symbols 0
Enter code a
Enter Length 0
Enter No. Of Symbols 0
Enter code b
Enter Length 0
Enter No. Of Symbols 0
Enter code r
Enter Length 3
Enter No. Of Symbols 1
Enter code c
Enter Length 2
Enter No. Of Symbols 1
Enter code d
Enter Length 7
Enter No. Of Symbols 4
Enter code r
Enter Length 3
Enter No. Of Symbols 5
Enter code d
Decoded String Is  cabracadabrarrarrad
PS C:\Users\JOY PATEL\Desktop\Joy\DC Practicals> █
```

Conclusion: LZ77 decompression is an algorithm in which given input as triplets is converted to respective alphabetic pattern and final string as output is obtained