Question 7:

Python Code:

Python File 1: Prefix_free_code.py

```
import string

#Class definition of tree for generating a Prefix Free Code
class Tree:
def __init__(self, cargo, left=None, right=None):
self.cargo = cargo
self.left = left
self.right = right
self.freq = 0
self.code =''

def __str__(self):
return str(self.cargo)
def getLetter(self):
return self.letter
def setLetter(self,value):
self.letter=value

def getFreq(self):
return self.freq
def setFreq(self,value):
self.freq=value
def getCode(self):
return self.code
def setCode(self,value):
self.code=value

#Stack definition to sort and store the created trees/symbols
class Stack:
def __init__(self):
self.items = []

def isEmpty(self):
return self.items == []

def push(self, item):
self.items.append(item)
```

```
def pop(self):
return self.items.pop()

def peek(self):
return self.items[len(self.items)-1]

def size(self):
return len(self.items)

#Sort and Store in Stack:
def sortedInsert(S, element):
if (S.isEmpty() or element.freq < S.peek().freq):
S.push(element)
else:
temp = S.pop()
sortedInsert(S, element)
S.push(temp)

#To visualise the tree:
def print_tree_indented(tree, level=0):
if tree == None: return
print_tree_indented(tree.right, level+1)
print ' ' * level + str(tree.cargo) + ' ' + str(tree.code)
print_tree_indented(tree.left, level+1)

#Code Assignment for the generated prefix free tree:
def update_code(tree, appcode):
if tree == None: return
update_code(tree.right, appcode+'1')
tree.code=tree.code+ appcode
update_code(tree.left, appcode+'0')

#Codebook Creation for encoding:
def create_codebook(tree):
if tree == None: return
create_codebook(tree.right)
if len(tree.cargo)==1:
wf.write(tree.cargo+' , '+tree.code+'\n')
create_codebook(tree.left)
#File Handling: Open and Reading the text in the file:
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Input
Text.txt','r')
para=rf.read()
rf.close()

#Length of the Paragraph read:
```

```
length_para=float(len(para))

#Creating a dictionary:
dictionary= string.ascii_lowercase + string.ascii_uppercase + string.punctuation + ' '

#Relative Frequency of each symbol:
codebook=[];

for i in range(len(dictionary)):
letter_count=string.count(para,dictionary[i])
codebook.append([dictionary[i],letter_count])

sym_tocode=Stack()

#Stack of all alphabets and punctuations with non zero frequency in a sorted manner
for element in codebook:
if element[1]!=0:
newNode=Tree(element[0]);
newNode.freq=element[1]
sortedInsert(sym_tocode, newNode)

#Prefix free code tree generation
while sym_tocode.size()>1:
t1=sym_tocode.pop()
t2=sym_tocode.pop()
mytree=Tree('(N)', t2, t1)
mytree.freq=t1.freq+t2.freq
sortedInsert(sym_tocode, mytree)
print_tree_indented(sym_tocode.peek())

#Code generation for the generated tree
update_code(sym_tocode.peek(), '')

#Generating Prefix Free Code Documentation for the provided text:
wf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Prefix Free
Code.txt','w')
create_codebook(sym_tocode.peek())
wf.close()
```

Python Code 2: encode_huffman.py

```
import string

#File Handling: Open and Reading the input text in the file:
```

```
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Input
Text.txt','r')
para=rf.read()
rf.close()

#File Handling: Refering to the Prefix Free Code Lookup Table for encoding
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Prefix Free
Code.txt','r');
codestring = rf.read().split('\n')
rf.close()

codebook = []
for i in range(len(codestring)-1):
codebook.append([codestring[i].split(' , ')[0], codestring[i].split(' , ')[1]])

#Generating Encoded Text:
wf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Encoded
Text.txt','w')
for ele in para:
for i in range(len(codebook)):
if ele == codebook[i][0]:
wf.write(codebook[i][1])
wf.close()
```

Python Code 3: decode_huffman.py

```
#File Handling: Refering to the Prefix Free Code Lookup Table for encoding
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Prefix Free
Code.txt','r');
codestring = rf.read().split('\n')
rf.close()

#File Handling: Refering to the Prefix Free Code Lookup Table for encoding
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Encoded
text.txt','r');
en_text = rf.read()
rf.close()

codebook = []
for i in range(len(codestring)-1):
codebook.append([codestring[i].split(' , ')[0], codestring[i].split(' , ')[1]])
codebook=sorted(codebook, key=lambda x: len(x[1]))

index = 0
```

```
wf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Decoded
Text.txt','w')
while index<len(en_text):
for i in range(len(codebook)):
if en_text[index:index+len(codebook[i][1]):1] == codebook[i][1]:
wf.write(codebook[i][0])
index += len(codebook[i][1])
break
wf.close()

#File Handling: Open and Reading the input text in the file:
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Input
Text.txt','r')
para=rf.read()
rf.close()

#File Handling: Open and Reading the input text in the file:
rf=open('D:\Documents\UB Course\Subjects\ITC\Homework\Exercise 4\Decoded
Text.txt','r')
de_para=rf.read()
rf.close()

#Error comparison between the input text and decoded text:
if de_para == para:
print 'Good'
else:
print 'Bad'
```

Output:

Input Text.txt:

Romeo and Juliet is a tragedy written by William Shakespeare early in his career about two young star-crossed lovers whose deaths ultimately reconcile their feuding families. It was among Shakespeare's most popular plays during his lifetime and, along with Hamlet, is one of his most frequently performed plays. Today, the title characters are regarded as archetypal young lovers.

Prefix Free Code.txt:

l , 1111
I , 11101111
J , 1110111 0
R , 1110110 1
T , 1110110 0
w , 111010
m , 11100
e , 110

o , 1011
i , 1010
d , 10011
y , 10010
s , 1000
t , 0111
W , 0110111
' , 0110111 0
, , 0110110
f , 011010
h , 01100
r , 0101
n , 01001
. , 0100011
- , 0100010 1
b , 01000100
c , 010000
  , 001
a , 0001
g , 000011
p , 000010
k , 00000111
v , 00000110
S , 00000101
q , 000001001
H , 000001000
u , 000000

<u>Encoded Text.txt:</u>

1110110110111100110101100100010100110011001110111000000011110101100111001101
010000010001001011101010001000011101001100100011101001011010011011111001001
0010100010010010001011011110101111111101000011100001000001010110000010000011
1110100000001011000010101110001100001010111110010001101001001001011001010100 0
0010100000001010111011001010010001010001001011000000011001011111010101100110 0
1010110000000100100001100110000110001010101000101010000010110110001000110100 1
1001111110110000011011001011000001111010011001011100011000110011100001011101 10
01000001000000111101111010110000010111110111100100010101100100001011010010 10
0001010111110001011101100110101001010010110101100000001001110100100100001100 10
1101000011100101011110101101000010001100111011101110011101000011000001000 1
1110010110100100001100100000101011000001000001111101000000010110000101011100 110
1110100000111100101110000111001000010101100001000000011110001010100100001011110
0011001010000011001100000000101101001001000011001011001010100000111110100110101
1001111010111001100010001010011001101101100010001111110110100100001100111101010
1001110110000100000100000011110011111001110110110001101010000011011010011100 01
1011011010001011001010100000111100101110000111001011010010111000000100100000011

00I00I0IIIIIIII00I000I0000I0II00I0I0II0I0I0II0I0IIII00II0I00II00I0000I0IIII000I
I00I0I0000I000II00IIII0II00I0III00II000II00I00II0II000I0IIII0II00II000I0IIII0I00
IIIIIIII1000I0I00000II00000I0I0I000I0I00000IIII100I0II00000I000I0I0IIII000I0I0II
I00000II000I0I0II00IIII0I00II00I000II00000I000I0I0I0I00000II00II00IIII00I000000I
0000IIII100II00I0I0II0000000I00I0000II00IIIIII0II00000II0II00I0II0000I0001I

Decoded Text.txt:

Romeo and Juliet is a tragedy written by William Shakespeare early in his career about two young star-crossed lovers whose deaths ultimately reconcile their feuding families. It was among Shakespeare's most popular plays during his lifetime and, along with Hamlet, is one of his most frequently performed plays. Today, the title characters are regarded as archetypal young lovers.