```python
''' Nadhem Ben Ameur             CIN: 11114647
    Anoir Feki                   CIN: 11113448
    Wissem Regaieg               CIN: 11104430
    Groupe: GI S3
    '''


def statistiques(texte):
        table = {}              #(un dictionnaire de la fréquence d'apparition de chaque caractère)
        for caractere in texte:
            if caractere in table:
                table[caractere] = table[caractere] + 1
            else:
                table[caractere] = 1
        L = list(table.items())
        L = [[x[0],x[1]] for x in L]
        return L                #liste de la fréquence d'apparition de chaque caractère

def inserefreq(x,liste):
        #x est une liste de deux élements: le caractère et sa fréquence d'apparition
        if liste==[]:
            return [x]
        elif (x[1])<=(liste[0][1]):
            return [x] + liste
        else:
            return[liste[0]] + inserefreq(x,liste[1:len(liste)])    #insere x dans la liste triee

def insereListe(liste1,liste2):
        if liste1==[]:
            return liste2
        elif liste2==[]:
            return liste1
        else:
            return insereListe(liste1[1:len(liste1)],inserefreq(liste1[0],liste2)) #insere liste1 dans liste2 de façon triee

def triefreq(liste):
        #renvoie la liste classée par ordre croissant de fréquence d'apparition des éléments de la liste des statistiques (Tri par insertion recursive)
        n=len(liste)
        if n==0 or n==1:
            return liste
        else:
            return insereListe(triefreq(liste[0:n//2]),triefreq(liste[n//2:n]))

def feuille(liste):
    return str(liste[0]).isalpha() #test booleen si l'arbre est une feuille ou pas

def poids(arbre):
    if arbre[0]:
        if feuille(arbre):
                return arbre[1]
        else:
                return arbre[0]         #retourne le poid d'une arbre contenant au moins une feuille

def fusion(arbre1 , arbre2):
        if arbre1 == [] :
                if arbre2 == []:
                        X=[]
                else:
                        X=arbre2
        elif arbre2 == []:
                if arbre1 == []:
                        X=[]
                else:
                        X=arbre1
        else:
                p = poids(arbre1)+poids(arbre2)
```

```python
                X = [p]+[arbre1]+[arbre2]
        return X                        #fusionner deux arbres


def arbre_codage(liste_freq_triee):
    arbre=[]
    for c in liste_freq_triee:
        arbre=fusion(c,arbre)
    return arbre                        #retourne l'arbre de huffman d'une liste triee suivant les frequences

def construit_code(arbre_huffman):
        liste_codage=[]
        t=(0,1)
        i=0
        suffixe=''
        while not feuille(arbre_huffman):
            char = arbre_huffman[1][0]
            liste_codage.append([char,suffixe+str(t[i%2])])
            i+=1
            suffixe+=str(t[i%2])
            i+=1
            arbre_huffman=arbre_huffman[2:][0]
        char = arbre_huffman[0][0]
        liste_codage.append([char,suffixe])
        return liste_codage                     #retourne la liste des caracteres ponderes par leurs codes binaires

def codage(code,phrase):
        dico={x[0]:x[1] for x in code}          #dictionnaire des mots codes
        msg_code=''
        for c in phrase:
            msg_code+=dico[c]
        return msg_code                         #genere le message compresse

def decodage(arbre_huffman,msg_code):
            liste_code = construit_code(arbre_huffman)
            dico_inverse={x[1]:x[0] for x in liste_code}   #dictionnaire dont les cles sont les mots codes et les valeurs sont les cararcteres
            codepart = ''
            phrase=''
            for c in msg_code:
                    codepart += c
                    if codepart in dico_inverse.keys():
                            phrase+=dico_inverse[codepart]
                            codepart = ''
            return phrase


def compresser():
        phrase = input("Entrer une liste de caracteres a coder :\n")
        print("phrase :\n{}".format(phrase))
        L = statistiques(phrase)                          #liste des statistiques
        print("liste des frequences :\n{}".format(L))
        L=triefreq(L)                                      #liste L triee par ordre croissant des frequences
        global arbre
        arbre = arbre_codage(L)                     #arbre de huffman
        print("arbre :\n{}".format(arbre))
        code = construit_code(arbre)                      #liste des mots codes
        print("code :\n{}".format(code))
        message_code=codage(code,phrase)
        print("phrase codee :\n{}".format(message_code))

def decompresser():
        phrase = input("Entrer un message a decoder :\n")
        print("message compresse :\n{}".format(phrase))
        code = construit_code(arbre)                      #liste des mots codes
        print("code :\n{}".format(code))
        message_decode=decodage(arbre,phrase)
        print("phrase decodee :\n{}".format(message_decode))

```

```
#TEST
'''
>>> compresser()
Entrer une liste de caracteres a coder :
abracadabra

phrase :
abracadabra

liste des frequences :
[['a', 5], ['b', 2], ['r', 2], ['c', 1], ['d', 1]]

arbre :
[11, ['a', 5], [6, ['r', 2], [4, ['b', 2], [2, ['d', 1], ['c', 1]]]]]

code :
[['a', '0'], ['r', '10'], ['b', '110'], ['d', '1110'], ['c', '1111']]

phrase codee :
01101001111011100110100

>>> decompresser()
Entrer un message a decoder :
01101001111011100110100

message compresse :
01101001111011100110100

code :
[['a', '0'], ['r', '10'], ['b', '110'], ['d', '1110'], ['c', '1111']]

phrase decodee :
abracadabra
>>>
'''
#======================================================================================================
'''
>>> compresser()
Entrer une liste de caracteres a coder :
ultrasonic

phrase :
ultrasonic

liste des frequences :
[['u', 1], ['l', 1], ['t', 1], ['r', 1], ['a', 1], ['s', 1], ['o', 1], ['n', 1], ['i', 1], ['c', 1]]

arbre :
[10, ['c', 1], [9, ['i', 1], [8, ['n', 1], [7, ['s', 1], [6, ['o', 1], [5, ['l', 1], [4, ['u', 1], [3, ['t', 1], [2, ['r', 1], ['a', 1]]]]]]]]]]

code :
[['c', '0'], ['i', '10'], ['n', '110'], ['s', '1110'], ['o', '11110'], ['l', '111110'], ['u', '1111110'], ['t', '11111110'], ['r', '111111110'], ['a', '111111111']]

phrase codee :
111111011111011111110111111110111111111111011110110100

>>> decompresser()
Entrer un message a decoder :
111111011111011111110111111110111111111111011110110100

message compresse :
111111011111011111110111111110111111111111011110110100

code :
[['c', '0'], ['i', '10'], ['n', '110'], ['s', '1110'], ['o', '11110'], ['l', '111110'], ['u', '1111110'], ['t', '11111110'], ['r', '111111110'], ['a', '111111111']]
```

```

phrase decodee :
ultrasonic
>>>
'''
```