ΑΣΚΗΣΗ

Να μελετηθεί και να εκτελεστεί ο παρακάτω κώδικας που υλοποιεί την κωδικοποίηση και αποκωδικοποίηση Huffman.

```
import sys

codes = {}

def frequency (str):
   freqs = {}
   for ch in str :
      freqs[ch] = freqs.get(ch,0) + 1
   return freqs

def sortFreq (freqs) :
   letters = freqs.keys()
   tuples = []
   for let in letters :
      tuples.append((freqs[let],let))
   tuples.sort()
   return tuples

def buildTree(tuples) :
   while len(tuples) > 1 :
      leastTwo = tuple(tuples[0:2])         # get the 2 to combine
      theRest  = tuples[2:]                 # all the others
      combFreq = leastTwo[0][0] + leastTwo[1][0]     # the branch points freq
      tuples   = theRest + [(combFreq,leastTwo)]     # add branch point to the end
      tuples.sort(key=lambda tup: tup[0], reverse=False)  # sort it into place
   return tuples[0]           # Return the single tree inside the list

def trimTree (tree) :
    # Trim the freq counters off, leaving just the letters
   p = tree[1]                              # ignore freq count in [0]
   if type(p) == type("") : return p            # if just a leaf, return it
   else : return (trimTree(p[0]), trimTree(p[1])) # trim left then right and recombine

def assignCodes (node, pat='') :
   global codes
   if type(node) == type("") :
        codes[node] = pat               # A leaf. set its code
    else  :                         #
        assignCodes(node[0], pat+"0")    # Branch point. Do the left branch
        assignCodes(node[1], pat+"1")    # then do the right branch.

def encode (str) :
    global codes
    output = ""
    for ch in str : output += codes[ch]
    return output

def decode (tree, str) :
    output = ""
    p = tree
    for bit in str :
        if bit == '0' : p = p[0]     # Head up the left branch
        else          : p = p[1]     # or up the right branch
        if type(p) == type("") :
            output += p              # found a character. Add to output
            p = tree                 # and restart for next character
    return output

def main () :
    debug = True
    str_init = sys.stdin.read()
    str = str_init[0:len(str_init)-1]
    freqs = frequency(str)
    tuples = sortFreq(freqs)

    tree = buildTree(tuples)
    if debug : print("Built tree", tree)

    tree = trimTree(tree)
    if debug : print("Trimmed tree", tree)

    assignCodes(tree)
    if debug : print(codes)

    small = encode(str)
    original = decode (tree, small)
    print("Original text length", len(str))
    print("Requires %d bits. (%d bytes)" % (len(small), (len(small)+7)/8))
    print("Restored matches original", str == original)

if __name__ == "__main__" : main()
```