Problem III. Implement and document Huffman algorithm in Haskell. Test your implementation on the example from figures 1-2 of week 3 file. Provide examples of encoding and decoding.

All programming targets were achieved to the 3rd degree. Below is proof of program correctness.

```
main = do
    -- Part 1 from Week 3 files
    let tree1 = makeCodeTree (makeLeaf 'a' 7) (makeCodeTree (makeCodeTree (makeLeaf 'b' 9) (makeLeaf 'c' 12)) (makeLeaf 'd' 22))
    -- Example of encoding
    let msg1 = "bcda"
    print "Original message:"
    print msg1
    let enc1 = encode msg1 tree1
    print "Encoded message:"
    print enc1
    -- Example of decoding
    let msg2 = [1, 0, 0, 1, 0, 1, 1, 1, 0]
    print "Expected encoded message:"
    print msg2
    let dec1 = decode msg2 tree1
    print "Decoded message:"
    print dec1
    -- Part 2 from Week 3 files
    let tree2 = makeCodeTree (makeCodeTree (makeLeaf 'd' 22) (makeLeaf 'e' 23))
                             (makeCodeTree (makeLeaf 'f' 27)
                                           (makeCodeTree (makeLeaf 'c' 12)
                                                         (makeCodeTree (makeLeaf 'a' 7) (makeLeaf 'b' 9))))
```

This screenshot shows our implementation working on the examples from the figures in the week 3 file.

The output can be seen below.

```
tux2: problem3 > ghci problem3.hs
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Huffman          ( problem3.hs, interpreted )
Ok, one module loaded.
*Huffman> main
"Original message:"
"bcda"
"Encoded message:"
[1,0,0,1,0,1,1,1,0]
"Expected encoded message:"
[1,0,0,1,0,1,1,1,0]
"Decoded message:"
"bcda"
"Original message:"
"abcdef"
"Encoded message:"
[1,1,1,0,1,1,1,1,1,1,0,0,0,0,1,1,0]
"Expected encoded message:"
[1,1,1,0,1,1,1,1,1,1,0,0,0,0,1,1,0]
"Decoded message:"
"abcdef"
"Original message:"
"fdbeca"
"Encoded message:"
[1,0,0,0,1,1,1,1,0,1,1,1,0,1,1,1,0]
"Expected encoded message:"
[1,0,0,0,1,1,1,1,0,1,1,1,0,1,1,1,0]
"Decoded message:"
"fdbeca"
*Huffman>
```