

# Homework #7

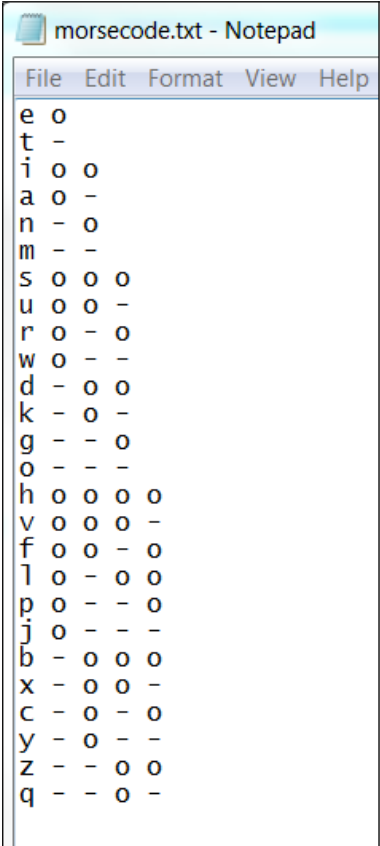
Assigned: 4/18/21

Due: 4/30/21 by 5 PM

This is the final assignment in the course. You should not underestimate it however, as there is quite a bit going on.

We will begin using a Tree to implement a solver for Morse Code. Morse Code is a simple encoding of the alphabet that uses dashes and dots to represent a single letter. The fact that more common letters use less symbols makes it an ideal candidate for a tree. You are heavily encouraged to use the Tree code presented in class.

First, you must build up a Tree. A screenshot of a text file defining the structure of your Tree is below:



```
morsecode.txt - Notepad
File Edit Format View Help
e o
t -
i o o
a o -
n - o
m - -
s o o o
u o o -
r o - o
w o - -
d - o o
k - o -
g - - o
o - - -
h o o o o
v o o o -
f o o - o
l o - o o
p o - - o
j o - - -
b - o o o
x - o o -
c - o - o
y - o - -
z - - o o
q - - o -
```

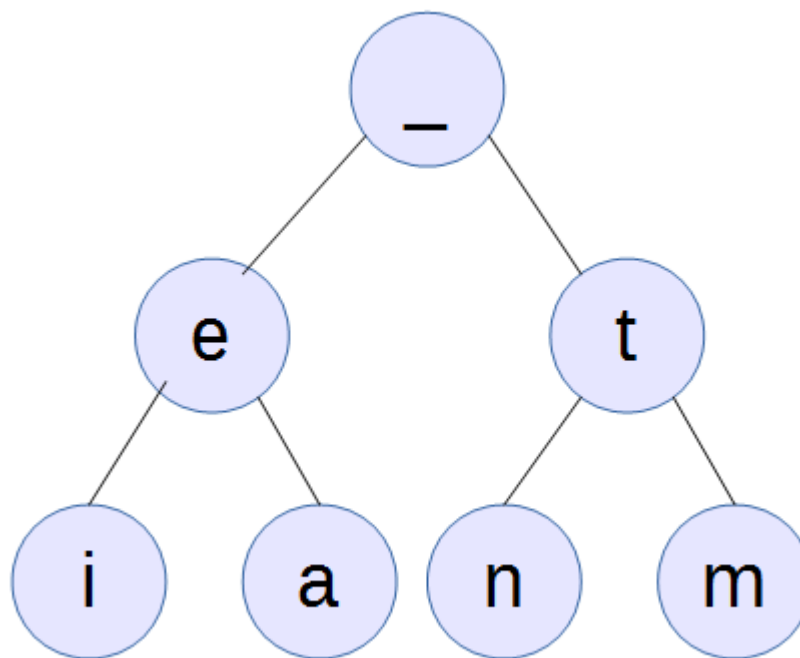
There are two ways you can populate the Tree:

- Use File I/O to read the file line by line and add it to an empty Tree. This is an advanced approach, only because we never covered File I/O in class, so don't feel obligated to do this. If you do go this route, use Scanner!
- Insert each of the letters into a Tree manually. This approach will require a fair amount of coding in the Main function, and you will need to be careful not to insert a letter into the wrong position

Regardless of the approach you choose to use, the following is how you will insert the letters into the Tree:

- The root is no letter, it is merely a starting point
- If a character is represented by a dot ( o ), it will be a left descendant
- If a character is represented by a dash ( - ), it will be a right descendant
- If a character is represented by one symbol, it will be one level below the root
- If a character is represented by N symbols, it will be N levels below the root

If done correctly, your Tree will resemble this image:



Note: The lower levels of the tree are not pictured

Prepare the following Class's

- **MorseTree:**
  - Contains an **TreeNode** representing the root of the Tree containing letters
  - Contains a default constructor to make a **null** Tree
  - Implement a method that will return a String containing the Pre-order of the Tree
  - Implement a method that will return a String containing the Post-order of the Tree
  - Implement a method that will translate an English string into its Morse code equivalent (ignoring case)
- **MorseTester:**
  - Contains a main function that will:
    - Instantiate a **MorseTree**
    - Fill the **MorseTree** in a manner of your choosing such that it contains the Morse Code (as defined above)
    - Produce the preorder of this Tree
    - Produce the postorder of this Tree
    - Define a string of English text, and translate it to Morse code using **MorseTree**

Example output: (note: you should not italicize your output, and ellipsis are used only to indicate that more is expected)

Your pre/post order functions should return a string with each letter separated by a space

*Preorder tree contents: e i s h v ...*

When translating from English to Morse code, the pipe character ( | ) separates each character. Whitespace is totally ignored.

*Input: The quick fox*

*Output: - | o o o o | o | - - o - | o o - | o o | - o - o | - o - | o o - o | - - - | - o o - |*

Be careful, this assignment only has one Class, and a tester, you need to write, but it is a **deep** one. Much work will be required for each of its methods!

Submit .java files for each of the above to the Brightspace link for Homework #7