Assumptions:

Of note, I have assumed the following when creating the translator:

* When a person uses this translator, they will not intentionally attempt to put in incorrect information, such as entering in an invalid character.
* If someone enters invalid information, it will be ignored as a result.
* There will not be anything deeper than 4 levels, or less than 4 levels. This is because we are dealing with the alphabet, with 26 letters, which needs 4 levels.
* The program doesn’t care about what the morse code information input through the file is, so long as each level has no more than the maximum, i.e Level 1 has 2 letters, Level 2 has 4, Level 3 has 8, and Level 4 has 16. For this reason, there can be any amount below the maximum for each level, so long as there are still 4 levels and that everything is used.
* In order to check to see if it is working or not, I included an option to see the in order version of the Binary Search Tree for the translator, with the top representing the leftmost while the bottom represents the rightmost. I also assumed that in order to double check to see if the information was input correctly, I let the viewers see the key and morse key information
* In order to give a blank root, I made the assumption that a blank one must be created and made the original root. It isn’t affected by anything and doesn’t affect anything itself outside of making the binary search tree as per the request.
* The information in the given file will be in the order of letter immediately followed by the morse code version, in . and \_ format.

Raymond Rennock

UML: (There is one class, one struct and the rest has been put into the main file)

|  |
| --- |
| **Letters** |
| - character: char  - morse: string  - size: int |
|  |

|  |
| --- |
| **Struct** |
| - key: Letters  - left: Struct  - right: Struct |
| + \*newNode(item: Letters): struct node  + insert (node: struct node\*, key: Letters, deep: int): struct node\*  + inorder(\*root: struct node): void  + letter\_to\_morse(root: struct node\*, letter: char): struct node\*  + morse\_to\_letter(root: struct node\*, word: string): struct node\* |

Big O:

struct node \*newNode(Letters item)= O(1)

struct node\* insert(struct node\* node, Letters key, int deep) = O(N)

void inorder(struct node \*root) = O(N)

struct node\* letter\_to\_morse(struct node\* root, char letter) = O(N)

struct node\* morse\_to\_letter(struct node\* root, string word) = O(N)

int input(int num, int lower, int upper) = O(N)