**Colin McNish**

**CS 303**

**References**: I did **not** build the logic for the base binary tree (Binary\_Tree and BTNode) as those were already provided in our class project files. The functions I created were **encode,decode, and create\_empty\_tree**

**Assumptions:**

* The project does not specify how to build the tree (recursion, build from an array etc) so I opted to build the tree manually rather than use any for loops or recursion as both those approaches resulted in a lot of confusion and failed implementations. This was a little tedious but manageable as the tree only has 27 nodes.
* The project does not specify how the messages should be encoded either, simply stating “Encode a message by replacing each letter by its code symbol” so I opted to use a vector of pairs <char,string> to have each morse code translation stored with its appropriate character. The decode function uses the process specified of traversing through the binary tree to decode the message.

**Big O:**

* Create\_empty\_tree(): This function is not reliant on data size so should be constant time O(1)
* Encode(): Most complex thing this function does is iterate through a vector of pairs so should be O(n)
* Decode(): essentially iterating through a binary tree and the user\_string char by char. Iterating through binary tree is roughly logn and iterating through a string is O(n) so iterating through the string here takes priority so performance is O(n).

Basic Overview of program:

* This program essentially just adds 3 new functions to the Binary\_Tree class to accomplish this morse encoding and decoding.
  + create\_empty\_tree(): This function creates a binary tree of depth 4 with all of the nodes in their appropriate positions with their data value set to the appropriate alphabet character
  + Encode: Encodes the user string to morse code by looking through our vector of pairs and replacing that character with its appropriate morse equivalent.
  + Decode: Reads morse code by starting at the root of the binary tree and going left when encountering a “\*” and right when encountering a “-”, resetting back to the root after each word of the user\_phrase.

Improvement that can be made:

* The output is still a little finicky as it does not output spaces between each word when decoding and entering a string of morse code directly into the decode function does not function correctly as the spacing which the function relies on is not consistent when copy pasting from say a word document to visual studio.

UML diagram:

