Data structures:
We implemented a binary tree that uses nodes. Each node contains a char* word and two nodes left and right.

Error cases:
The program ensures there is exactly one string received. Any other case returns 0.

Setup:
We initialize two integers, j and start, that will serve as our indexes. Start will represent the start index of a word and j will represent the end index of a word.

We proceed to iterate through every character in the input. There are two cases to consider.
First case(1):
If the current character is a letter, we simply increment j++.
Second case(1):
If the character is not a letter, it is a separator.

In the second case we must check the length of the word.
First case(2):
If j - start is equal to 0, that means we begin with a separator or there are multiple separators in a row. We ignore the separator by incrementing j and start by one.

Second case(2):
If j - start is greater than 0 then we have a word of at least length one. We will take steps to copy the word out of the input and insert it into our binary tree

1. Malloc the length of the word (j- start) and add one to it for the null terminator into a new char* [s].
2. Use strncpy to copy the first j-start characters from the input string starting at index start
3. Since strncpy does not add the null terminator, we add it to the end of our word
4. Create a new node with the word and insert it into our tree.
5. We set start to j + 1 to mark the possible beginning of the next word and then increment j by one

At the end of the iteration if there is a word at the end of the input that we did not insert we repeat second case(2) from 1 - 4.