Kesong Xie
Refactor Code

1.
Get rid of the unnecessary <regex> library included in the header while the file does not use it at all.

2.
Added consistent comments for each function, including the its general description, and required parameters and return values.
For example, the parameter is labeled with @param in the comment, and the return is labeled with @return

```
/* DictionaryTrie::createNode: a convenient helper function for
creating a node
 * @param word: the word that the node created from
 * @param index: the index for the char in the word, that is used for
the data for the node
 * @param freq: the frequence for the word, this value is used only
when the char at the given index is the last char of the word
 * @return Node*: the node being created
 */
Node* DictionaryTrie::createNode(std::string word, unsigned int index,
unsigned freq){
    /*default false if this is not the last char*/
    bool isWordEnd = false;
    /*default 0 if this is not the index for the last char*/
    unsigned frequence = 0;
    if(index == word.length() - 1){
        /*this is the last char*/
        isWordEnd = true;
        frequence = freq;
    }
    return new Node(word[index], isWordEnd, frequence);
}
```

3.
Remove unnecessary output from std::cout from the debugging process.

4.
Remove redundant white spaces

5.
Group related function together. For example, the destructor will call the a recursive function for deleting all the dynamically allocated memory, they should be group together. See below that postOrderDeleteAux is just below the DictionaryTrie

```
/** DictionaryTrie::~DictionaryTrie: the destructor for the
DictionaryTrie class, which is repsonsible
 *  for deallocated any dynamically memory associated with the class.
This destructor would call the
 *  recurssive postOrderDeleteAux for cleaning up the dynamically
allocated memory
 */
DictionaryTrie::~DictionaryTrie(){
    postOrderDeleteAux(root);
}



/* postOrderDeleteAux: post order deallocate memory for nodes
 * @param Node* node: the starting node for post order traversal
deletion
 * @return void
 */
void DictionaryTrie::postOrderDeleteAux(Node* node){
    if(node != NULL){
        postOrderDeleteAux(node->left);
        postOrderDeleteAux(node->middle);
        postOrderDeleteAux(node->right);
        delete node;
    }
}
```

6. Added inline comments inside the function body to help organize the responsibility of the specific block of code.