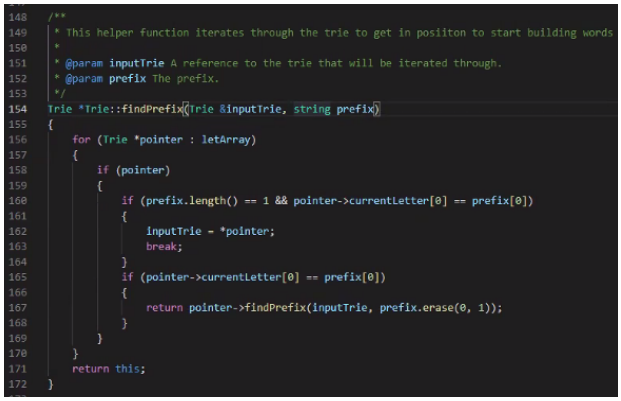
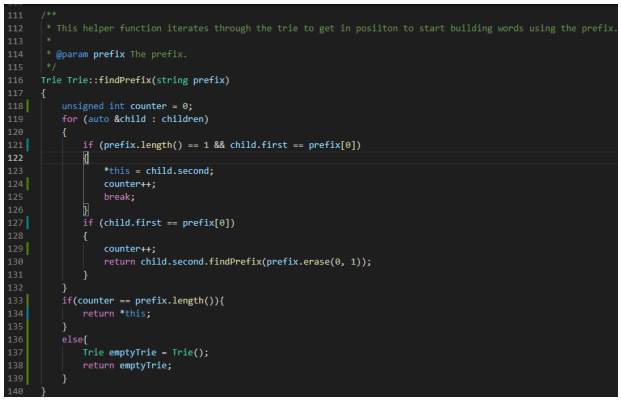
Harry Kim
Braden Morfin

Refactoring A4

One of the big refactorings we did with our Trie.cpp was using map in place of an array. The most significant change that we made was with the helper method findPrefix for the allWordsStartingWithPrefix. Since we did not work with an array of pointers anymore, we no longer needed to check if a pointer was a nullptr in the array. All we needed to do was iterate through the existing value pairs of the map. If it didn't exist, it would simply not be within the map.

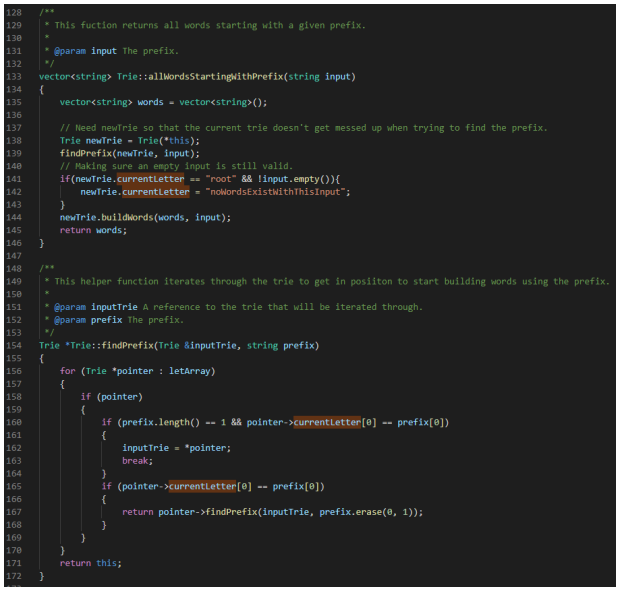


```
/**
 * This helper function iterates through the trie to get in posiiton to start building words
 *
 * @param inputTrie A reference to the trie that will be iterated through.
 * @param prefix The prefix.
 */
Trie *Trie::findPrefix(Trie &inputTrie, string prefix)
{
    for (Trie *pointer : letArray)
    {
        if (pointer)
        {
            if (prefix.length() == 1 && pointer->currentLetter[0] == prefix[0])
            {
                inputTrie = *pointer;
                break;
            }
            if (pointer->currentLetter[0] == prefix[0])
            {
                return pointer->findPrefix(inputTrie, prefix.erase(0, 1));
            }
        }
    }
    return this;
}
```



```
/**
 * This helper function iterates through the trie to get in posiiton to start building words using the prefix.
 *
 * @param prefix The prefix.
 */
Trie Trie::findPrefix(string prefix)
{
    unsigned int counter = 0;
    for (auto &child : children)
    {
        if (prefix.length() == 1 && child.first == prefix[0])
        {
            *this = child.second;
            counter++;
            break;
        }
        if (child.first == prefix[0])
        {
            counter++;
            return child.second.findPrefix(prefix.erase(0, 1));
        }
    }
    if(counter == prefix.length()){
        return *this;
    }
    else{
        Trie emptyTrie = Trie();
        return emptyTrie;
    }
}
```

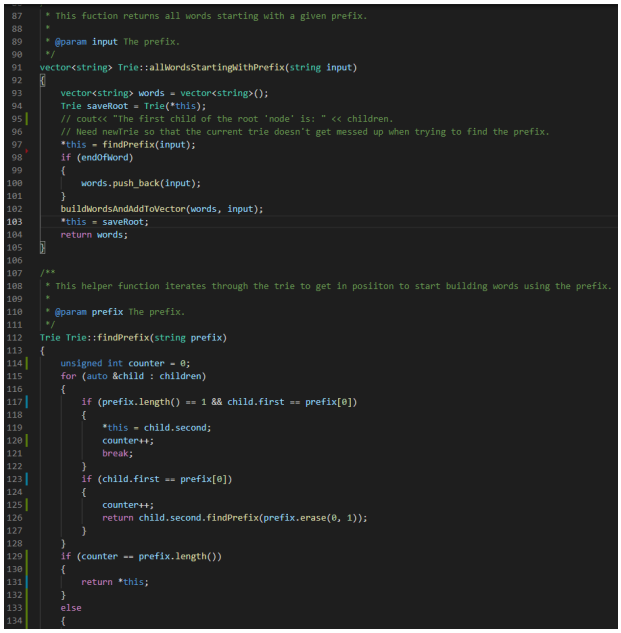
The second biggest refactor we made was getting rid of the member variable currentLetter. This variable was a string that stored the letter of the current Tire (node), and in the case of the root was simply just "root". We got rid of this because it wasn't needed because in the map the key was the current letter, so to store it in a variable is redundant. A lot of our methods relied on currentLetter, so we had to go through and make changes in a lot of places. Most being simple changes from "currentLetter" to the key of the value. However, reimplementing the allWordsStartingWithPrefix method took a while to redesign without using currentLetter. As representing the root node took some clever thinking.



```
/**
 * This fuction returns all words starting with a given prefix.
 *
 * @param input The prefix.
 */
vector<string> Trie::allWordsStartingWithPrefix(string input)
{
    vector<string> words = vector<string>();

    // Need newTrie so that the current trie doesn't get messed up when trying to find the prefix.
    Trie newTrie = Trie(*this);
    findPrefix(newTrie, input);
    // Making sure an empty input is still valid.
    if(newTrie.currentLetter == "root" && !input.empty()){
        newTrie.currentLetter = "noWordsExistWithThisInput";
    }
    newTrie.buildWords(words, input);
    return words;
}

/**
 * This helper function iterates through the trie to get in posiiton to start building words using the prefix.
 *
 * @param inputTrie A reference to the trie that will be iterated through.
 * @param prefix The prefix.
 */
Trie *Trie::findPrefix(Trie &inputTrie, string prefix)
{
    for (Trie *pointer : letArray)
    {
        if (pointer)
        {
            if (prefix.length() == 1 && pointer->currentLetter[0] == prefix[0])
            {
                inputTrie = *pointer;
                break;
            }
            if (pointer->currentLetter[0] == prefix[0])
            {
                return pointer->findPrefix(inputTrie, prefix.erase(0, 1));
            }
        }
    }
    return this;
}
```



```
 * This fuction returns all words starting with a given prefix.
 *
 * @param input The prefix.
 */
vector<string> Trie::allWordsStartingWithPrefix(string input)
{
    vector<string> words = vector<string>();
    Trie saveRoot = Trie(*this);
    // cout<< "The first child of the root 'node' is: " << children.
    // Need newTrie so that the current trie doesn't get messed up when trying to find the prefix.
    *this = findPrefix(input);
    if (endOfWord)
    {
        words.push_back(input);
    }
    buildWordsAndAddToVector(words, input);
    *this = saveRoot;
    return words;
}

/**
 * This helper function iterates through the trie to get in posiiton to start building words using the prefix.
 *
 * @param prefix The prefix.
 */
Trie Trie::findPrefix(string prefix)
{
    unsigned int counter = 0;
    for (auto &child : children)
    {
        if (prefix.length() == 1 && child.first == prefix[0])
        {
            *this = child.second;
            counter++;
            break;
        }
        if (child.first == prefix[0])
        {
            counter++;
            return child.second.findPrefix(prefix.erase(0, 1));
        }
    }
    if (counter == prefix.length())
    {
        return *this;
    }
    else
    {
```