Justin Francis and Palani Thangaraj
9/26/19
CS 3505

A4 Write Up

Assignment 4’s objective was to refactor A3 inorder to make it more efficient and easier to maintain, by using a map instead of a C-style array containing pointers to objects. The original code is full of pointers, addresses, and arrow operators. Before refactoring, it was a dismal mess of syntactic gymnastics such as,

```
if(i == prefix.length() - 1 && node -> character[letter]->isEnd)
```

After refactoring, the code lost much of the syntactic fog and became simpler to understand. The following is the same line of code after refactoring,

```
if(currentNode->isEndOfWord)
```

Another reason to switch to a map was the ease of implementing the Rule-of-Three. Map is a supported C++ class so it has its own implementation of the Rule-Of-Three, making code maintenance significantly less. The original Rule-of-Three,

```
Trie::Trie()
{
  for(int i = 0; i < ALPHABET_SIZE; i++)
  {
    alph_[i] = nullptr;
  }
  isEndOfWord = false;
}
Trie::Trie(const Trie& toCopy)
{
  isEndOfWord = toCopy.isEndOfWord;

  for(int i = 0; i < ALPHABET_SIZE; i++)
  {
    alph_[i] = nullptr;
    if(toCopy.alph_[i])
    {
      alph_[i] = new Trie(*toCopy.alph_[i]);
    }
  }
}
Trie::~Trie()
{
  for(int i = 0; i < ALPHABET_SIZE; i++)
```

Justin Francis and Palani Thangaraj
9/26/19
CS 3505

```
  {
    delete alph_[i];
  }
}
Trie& Trie::operator=(Trie other)
{
  std::swap(alph_, other.alph_);
  std::swap(isEndOfWord, other.isEndOfWord);

  return *this;
}
```

Refactored Rule-of-Three:

```
Trie::Trie()
{
  isEndOfWord = false;
}
```

The code complexity decreased, the smells decreased, and the readability was improved.

Another improvement from this round of refactoring was variable naming. Obscure data members and local variables with single character names such as

```
alph_[c];
```

to a much more descriptive names.

```
alphabetMap.find(letterInWord);
```

Another example is for loop variables.

1) `for(std::string::iterator iter = word.begin(); iter != word.end(); ++iter)`
2) `for(int i = 0; i < word.length(); i++)`

To

1) `for(auto currentNode : alphabetMap)`
2) `for(auto letterInWord : word)`