CS32: Project 4

For this programming assignment, I implemented a linked list hash table, such that it is a list of Node pointers. Node is a struct I defined in order to link each data member in the form of a linked list. Thus, a Node upon creation is given a string ID and an int lineNumber and an int scope. This information is stored into private member variables mID, lineNum, and mScope, respectively. I chose to use the data structure of a hash table of a linked list as this would increase the efficiency, especially in relativity to the previous implementation.

The inefficient implementations of the functions were mostly linear, with the exception of enterScope(), which was constant time. My functions have the same time complexity, however, perform better by employing a hash function to search only desired identifiers that were declared in the same scope, and hashed to the same bucket. In this way, I reduced N by means of the items the functions have to search, instead of looping through the entire data set. Specifically, my exitScope() function is O(N) with N being the number of identifiers declared in the scope for intended deleting. The find and declare functions are O(N) with N being the number of items in a particular bucket that are declared in a certain scope.

The most challenging aspect of this assignment for me was testing the program. The utilization of the given generateTests.cpp was intimidating. However, after running the program with this file and with some trial and error, I realized the purpose of the file. With these generated test cases, I ran the program with the original main function, and passed with performance being within a factor of two of the optimal performance given by the spec.

***Pseudocode***

*SymbolTableImpl’s declare function*

bool SymbolTableImpl::declare(const string& id, int lineNum)

{

if id is empty

return false

hash passed string id to get bucket index

loop through the hash table of this bucket

if found return false

Create a Node pointer to a new Node

Push this Node to the front of the hash table

Push this Node onto the stack of Nodes

Return true

}

*SymbolTableImpl’s find function*

int SymbolTableImpl::find(const string& id) const

{

if id is an empty string

return -1

hash passed string id to get bucket index

loop through the hash table of this bucket

if found return line number

return -1

}

*SymbolTableImpl’s exit scope function*

bool SymbolTableImpl::exitScope()

{

if node stack is empty

return false

loop through node stack until empty or the top of the stack is entry marker “”

extract top Node

hash top Node’s id

loop through the hash table of this bucket

if iterator’s id is the top Node’s id

pop the stack

set Node pointer to iterator

set iterator to what erase function returns – the node that proceeds the erased node, which is what the iterator was pointing at, in the list

delete Node pointer

break

else

increment iterator

if scope count is zero

return false

else

pop the stack

decrement scope count

}