# CS 562
# Final Report

Bowen Yu

3/14/2016

**1. What I accomplished**

First of all, by doing this project, I have learned how to use Template Scripting Testing Language(TSTL) to test the code. Because TSTL was a totally new thing to me, I rarely know anything about TSTL. After learning it through this term, I knew the meaning of the functions and sentences of the TSTL, and then I learned how to write codes to test the program, and finally I am able to write write my own functions to help me to test the program more appropriately .I think it is the most important thing that I have accomplished in this term.

I used the TSTL to use a library which is comprehensive. I have done testing three programs, which are Single Linked List, Doubly Linked List, and Adjacency List. At the beginning of this term, I just used TSTL to test the insert function of those three programs. By the end of this term, I have tested most of the functions, including insert(), append(), returnIndex(), updateIndex(), insertBeforeIndex(), insertAfterIndex(), deleteData(), deleteAllData(), insertBeforeEveryData(), insertAfterEveryData() for singly linked list. For the doubly linked list, the functions that I tested are similar, the only differences are there are no updateIndex() and returnIndex(). For the adjacency list, I just tested the insert() function.

Insert() function: this function is used for adding new data to the head of the singly linked list and doubly linked list. In order to test it, I defined a function search(), and then use search to check whether the first integer is the integer added or not. And then I defined a function cap() to represent how many data in the singly linked list and doubly linked list. And then used it to check the the total number of the linked list, if the total number of linked list is added by 1, that means insert() function is called successfully, otherwise, it fails.

Append() function: this function is used for adding new data to the tail fo the singly linked list and doubly linked list. This function is quite similar with insert() function. First thing is to test if the last data is the same to the new data that added. And then check the number of the singly linked list and doubly linked list.

ReturnIndex() function: this function only exists in the singly linked list. In order to test it, it would divide into two parts: first, I must check when there is no data in the singly linked list, it is none. Second, after inserting a data, is the first index 0.

InsertBeforeIndex() function: this function is used for inserting a new data before the particular index. In order to test it, I need to determine the number of index should smaller than the total number of the linked list. For example, the total number of the index is 5, I need to make sure the index can be add should less than 5, if I want to add a data before the ninth index, it would not work. And then I used the search() function to check whether the data I added correctly. The final step is check whether the total number of the linked list was added by 1.

InsertAfterIndex() function: this function is used for inserting a new data after the particular index. The way of testing it is similar to the way of testing insertBeforeIndex() function. The only difference is “before” a index and “after” a index.

DeleteIndex() function: this function is used for deleting one index. First, I test whether the index I want to delete in the linked list. And then use copList() function to copy the linked list and compare in order to determine if the deletion complete. Finally the length of the linked list minus by 1.

DeleteData() function: this function is used for deleting one data. I used almost the same method to the DeleteIndex().

DeleteList() function: just test let the cap() of the linked list be 0.

## 2. Bug report

By the end of this term, I still find no bugs. I had searched a lot of library in the beginning of this term, this library is the most comprehensive and well organized library I found. However, there are some flows. For instance, if I use a function as a parameter for another function, the result would be “False”, but the function which is a parameter actually runs. I think this cannot count as a bug, but it is truly a small flow.

## 3. How well does the tester work?

I think the random tester of TSTL works quite well. The random tester can pick a mount of random numbers to help to test the python code. It is very helpful because the design a lot of random input is the most difficult part for testing.

## 4. What is wrong with TSTL?

The first thing is lack of the related information or document to guide the new users to use it. Second, for most users, they may prefer a interface of the software. Finally, it can only test python codes.

## 5. What is good with TSTL?

TSTL can provide random tester for testing the python function. When the user get used to the TSTL, they will find that the TSTL is very easy to use. The user do not need to install many related software, and also they do not need to set configurations. It is helpful for the users. TSTL has very high quality to find the bugs. It can show the coverage. I think it can be popular.

## 6. Coverage summary

The total coverage is 47% for the singly linked list. There are total 10000 operations of the test. The result is as following:

For the singly linked list:

```
47.6923076923 PERCENT COVERED
4.0595369339 TOTAL RUNTIME
100 EXECUTED
10000 TOTAL TEST OPERATIONS
3.41299295425 TIME SPENT EXECUTING TEST OPERATIONS
0.497232437134 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.0115330219269 TIME SPENT CHECKING PROPERTIES
3.42452597618 TOTAL TIME SPENT RUNNING SUT
0.048276424408 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
139 BRANCHES COVERED
105 STATEMENTS COVERED
```

The total coverage is 68% for the doubly linked list. There are total 10000 operations of the test. The result is as following:

For the doubly linked list:

```
68.2352941176 PERCENT COVERED
4.08042597771 TOTAL RUNTIME
100 EXECUTED
10000 TOTAL TEST OPERATIONS
3.34407567978 TIME SPENT EXECUTING TEST OPERATIONS
0.594952821732 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.0103557109833 TIME SPENT CHECKING PROPERTIES
3.35443139076 TOTAL TIME SPENT RUNNING SUT
0.0498375892639 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
151 BRANCHES COVERED
115 STATEMENTS COVERED
```