CS562 Winter Quarter 2016
Applied Software Engineering
Final Report

Qi Wang
ID: 932-439-151
wangq5@oregonstate.edu

Section 1. Description about what I accomplished
After completing this project, I learned how to use the Template Scripting Testing Language(TSTL) to test the python program. I use the library is created by the Michael Robertson. I tested his doubly linked list python program. In this program, there are 12 main functions, which includes the insert() function, append() function, deleteList() function and insertAfterEveryData() function etc.

At the end of this term, I finished testing most functions of this python file. In the tstl file, there are three main parts. The first part is utility functions, which are designed according to the testing purpose and testing functions. I designed fourteen utility functions. For example, the insertTest, appendTest, count the length of the list. The utility functions are used in the actions design. The second part of the tstl file is pools, which are where TSTL stores the state of the SUT. In third part of tstl file, it contains many actions.

The main functions of this doubly linked list program can be classified into two types: insert and delete. For the insert functions, they are mainly used to insert data to the list. So I finished testing the following functions of the python file:
Insert(self, data): this function can add a new data to the head of the doubly linked list.
In order to test the insert() function, I firstly get the length of the doubly linked list, and use the insertTest utility function to make sure the insert function is done correctly.
Append(self, data): this function could add a new data to the tail of the doubly linked list. This function’s testing method is similar with the insert function, firstly get the length of the doubly linked list to ensure after appending a data, the length of the linked list is added by one. Then use the appendTest function to ensure appending function is right.
insertAfterIndex(self, index, data): this function can insert a new data after the given index position of the linked list. In order to test this function, first should use the function to get the number of the data in the doubly linked list, then if the insertAfterIndex function works successfully, the length of the list will be the original length of the list plus the extra adding data’s number.
insertBeforeIndex(self, index, data): this function could insert a new data before the given index position of the linked list. The testing method is similar to the above one.
insertAfterEveryData(self, data, dataToInsert): this function is used to insert a new data after every data of the doubly linked list. In order to test this function, I created an utility function which is used to return the count of the data’s number of the doubly linked list. Then use this function to know the number of the list. Then get the length of the after inserting data, and ensure the length of list is correct.
insertBeforeEveryData(self, data, dataToInsert): this function is used to insert new data before every data of the doubly linked list. This function’s testing method is pretty the same as the insertAfterEveryData testing function. It also firstly use the defined function to get the count of the data’s number of the list. Then get the length

of the doubly linked list to ensure the length of the list is right.

For the delete types functions, I also test some main functions, which is shown in the following:
deleteList(self): this function can delete the list. In order to test it, I defined a isEmpty function in the first part of the tstl file. If the isEmpty function returns true, then the deleteList function is correct.
deleteData(self, data): this function can delete a existing data of the doubly linked list. When testing this function, I firstly created a function to check if the target value in the doubly linked list. Then, using return length to test if it delete successfully.
For the deleteIndex(self, index) function and deleteAllData(self, data) functions, they can also do the delete work, the first one can delete the given index position data and the second function can delete all the same data of the doubly linked list. These functions's testing is also similar to the deleteData testing function. So first of all checking if the targeting data in the list, then ensure after deleting the length of the list is correct.
So in the tstl file, I tested ten main functions and wrote fourteen utility helper functions.

Section 2. Bug report
At end of term, after testing the python program, I found the author is very thoughtful, and considers many situations and cases. So I did not find any bugs for this program when using the tstl file to test. However, I only found some not very well results when running the python file. For example, when running deleteData(a.insert(4)), it returns a false. It is supposed to return a false to show that the input attribute of data is wrong, but it actually done the insertion of number 4 twice to the doubly linked list. I think the print out result should be more specific and show the reason. However, this is not counted as a bug when using the tstl file to test the doubly linked list.

Section 3. How well does the tester work?
I think the random tester of TSTL is designed well. The random tester can help test a specific test case with the randomly input number. For example, for this doubly linked list program, it can randomly give the new data to deal with the insert or append functions. As long as the characters that are been designed in the file, no matter the type of the input character, it can help to test all possible inputs.For the doubly linked list designer, personally, I think he actually designs a very good doubly linked list. Even though when calling insert functions before calling the delete functions, it shows the result is not very clear, the author overall includes enough situations and possibility.

Section 4. Disadvantages of TSTL
To be honest, TSTL is very new for me. This is the first time know about this testing language. So there are some places that are very unfamiliar to me, such as the syntax. So I designed my project as the examples showing in the TSTL file. TSTL has

different syntax, sometimes an error shows up, it will take a very long time to figure out the reason or never. And sometimes when running the tstl file, it has a series of warning: No data was collected. Also when it finds a problem of the program, it will stop and never run the rest of the code. I personally think it is very goof for the new user to know that if there is a very specific document to tell and guide the user how to design the tstl file.

Section 5. What is good about TSTL?
For the good part of TSTL, it firstly provides a random tester for the testing. Users can save a lot of time on the input characters. It is actually a very good language to help the designers to find the bugs. It is also easy to install and use, and not very difficult for the users to learn. I think it could be very useful in the future.

Section 6. Coverage summaries, any other important data on the test effort
For this doubly linked list project, at the end of this term, the coverage is 72 percent covered. There are total 1000test operations. The following number is the screenshot of the running result.

```
71.1568627451 PERCENT COVERED
3.01875495911 TOTAL RUNTIME
100 EXECUTED
10000 TOTAL TEST OPERATIONS
2.60906672478 TIME SPENT EXECUTING TEST OPERATIONS
0.276334285736 TIME SPENT EVALUATING GUARDS AND CHOOSING ACTIONS
0.00923299789429 TIME SPENT CHECKING PROPERTIES
2.61829972267 TOTAL TIME SPENT RUNNING SUT
0.0515999794006 TIME SPENT RESTARTING
0.0 TIME SPENT REDUCING TEST CASES
155 BRANCHES COVERED
115 STATEMENTS COVERED
```