The Progress Report

CS562

Chao Zhang

Prof Groce Alex

This is a report of the progress of the test project. My test project is to test a library called LinkedList_Circular.py. In the beginning of the project, I thought this project is only related with the LinkedList_Circular.py file, but when I got start with it, I realized that this LinkedList_Circular.py is also connected with other two files, LinkedList_Single.py and LinkedList_Double.py. This made me changed my plan, I'm not only test the LinkedList_Circular.py but also test the other two files. For the other two files, the test in basically the same as the test for the Circular.py file. So I will focus on the Circular file to explain my test. In the last submission, my test already tested the check the length and check the insert. I added the check for the empty and the check for the append. In additional, I add the test for the return value. The first step for the test is building the pools to get all things ready. I have pool with 100 integers. For the tstl, I need to call the function I want to test and build the checker to check if all the actions this function made is correct. The test function looks like

```
# insert
        ~<LINKEDLIST>.insert(<INT>) => (leng(<LINKEDLIST,1>) ==
PRE<(leng(<LINKEDLIST,1>))>+1) and (checkInsert(%LINKEDLIST,1%,%INT%) == True )
```

This is the insert and length function tester. It will get an integer from the pool randomly and pass it the the insert function of the python file, after this insert, the function will check the length. The length after insert should be equal to the length before insert plus one. ~<LINKEDLIST>.insert(<INT>) this will pass the integer to the python file. (leng(<LINKEDLIST,1>) == PRE<(leng(<LINKEDLIST,1>))>+1) this will check the length. For this time, I added the function to check the append and return value. Those functions basically have the same structure as the insert function.

```
def checkAppend(self, D):
    ptr = self.head
    if(ptr == None):
      return False
    else:
      while ptr.next:
        ptr=ptr.next
      if (ptr.data == D):
       return True
      else:
       return False
```

I def the append like this. And this is the check append function:

```
# append
    ~%LINKEDLIST%.append(%INT%) => (length(%LINKEDLIST,1%) ==
PRE%(length(%LINKEDLIST,1%))%+1) and (checkAppend(%LINKEDLIST,1%,%INT%) == True )
```

In the further, I will test the updateIndex, deleteIndex, insertBeforeIndex, insertAfterIndex, deleteData, deletePtr, deleteAllData, insertAfterEveryData, insertBeforeEveryData, deleteList and many other functions.

For the deleteData() function, it is used for deleting specific the data in the list. After call this function, the value of the deleting data will not exist in the linked list, and the function will return a false. When it returns a false, it means the deleteData function is successful.

For the insertBeforeEveryData function. This test should call the insertBeforeEveryData function, and the added value should be in the beginning of the list, then the final length is the length of original list plus the number of new data.

And right now, I have not found any bug yet. The SUT is good, it is easy to use and can test the python file I chose perfectly for now. the author of this file have a strong logic and good at python. I will try to improve my test and try to find bugs in the further.

So far, I just test a little of this python file, I will try to test 90% of this file in the end of this term. I will work on the tstl, hope I can do better on it.