CS 562 Final Report

Yueting Zhu

Accomplishment

For this project, I suppose to test codes of converting between a string of ASCII and Unicode chars maintaining readability. I prepare to test this library which get from the GitHub and check if it can be used directly. In addition, I also want to test some functions about processing strings, which is discussed in preparing job hunting. However, at last, I finished several different codes on BST.

I tested three different kinds of BST code, two of them use key, and the one left does not. During these codes test, I find that the codes should be test individually not only because the ways to write the codes are different, but also the different definitions which may cause problem even though they are doing the same thing.

For the other code, string to rot Unicode, I just test a part of it because the unexpected performance of the result. The code is working on transferring strings to Unicode and according to operating the Unicode to let it can be read but different from the original string. For example, string 'Hello World!' to ‘Ĥèľľő Ŵőŕľđ!’. However, when I was testing the code, I find the result always has error, even I did a really simple test. I run the code and find that the result is not like what I expected. It appears as Unicode, not the readable words. To solve this problem, I need to find some

requirement codes for it. But I still cannot solve this problem. For this reason, I only tested part of this code, like safety function, string to Unicode function, and etc.

Bugs Report

In this project, I prepare to test codes of BST, string to rot Unicode, and other several small codes. I have already done the test of BST. In the process of testing BST, I tested delete, insert, length, repeat number processing, and different input types. I found some bugs in it. The first BST code, called BST1, cannot solve the multiple value with same key. Actually, if we insert a number with a key already exist, we should replace or process this problem. But the code did nothing but return an error. When I was testing, the test code was always stopped at that error. But if I use a large range of the random key size, it has less possibility to appear that error. In addition, the first one has one more bug, which is the code define the original key as the value input. The key should be used to judge the location of the value in the tree. So when I testing the code with string value of input, it returned an error. And some parts of code can be enhanced. According to the code, when I did delete test, the result of testing sometimes return error of ‘Empty Tree’. It is because if more delete happens, the tree I input will be empty and the code ban to delete anything from an empty tree and return error of that. However, in the code of delete function, I found that it allows

to delete an index from a tree, which does not contain the index, and return the original tree. To be the same, if using delete function on empty tree, it should return an empty tree not an error. I also pull out part of the original code to test if all of the code is necessary.

Other two codes are both good to be used. I tried to add some bugs, like delete the empty tree judgement part or change the length from -1 to +1 in the delete function, in those codes. At last, I found out the ‘bugs’ in it.

Evaluation of Tester Work

By using tester, we can find if there is bug or not. Normally, we use debug to test the code if there is any bug in it. But debug cannot tell whether the result is correct or not. It is hard to find the logical error. Using this test, we can know whether the code is correct or not, though the code is passed by running. But the problem is we need to think about the way that can test with full coverage of the code. This part is really hard but important. With a full coverage, all parts of the code would be checked. However, constructing full coverage tester is not easy. Tester can only test the correctness and show the errors in assertion, it cannot show the specific

location of the error in the code. At last, it still need programmer to analysis and find the error location. For these reasons, tester is good to be used to check the code, but still can be enhanced in the future.

Advantage of Tester

Tester can run the code on lots of different data to check if it is right. When we check the code correctness, we should test all kinds of possible inputs, or under different situations. Tester make it much easier to process a lot of database. A fully correct code is totally different with a most or nearly fully correct code. Unfortunately, the most codes we write are nearly fully correct. So tester is a good tool to make codes better.

Disadvantages of Tester

Tester cannot display the specific location of the error. It can be used to reduce the range of errors but not to tell what the error is. It still needs programmer to analysis and find errors. It is an easy way to help programmer check all cases in a short time. However, the error finding part is still based on programmer. And different code have nearly no similarities

in test code writing.

Coverage Summary

In the BST test, I use different methods in three kinds of BST codes.

Name	Stmts	BrPart	Cover
BST.py	135	6	25%
BST1.py	90	0	1%
BST2.py	184	7	26%
bubblesort.py	17	0	96%

Coverage is an important standard when we check the code correctness. In this project, because I split BST into three parts, so each one has low correctness, like more than 20%. The second one with 1% because some error in it which I cannot fix. The bubble sort code has the highest coverage, as it has less code, which is easy to be covered. As the performance problem has not been solved, the coverage of rot Unicode is also around 1%. I will still work on it and try to solve the problem.