**SE-576 Homework -1 (Blackbox and Whitebox testing)**

**Set-Up Steps:**

* Download the BlackboxTests.py, WhiteboxTests.py and binarySearch.py files into the Folder.
* Open the terminal/command line and Navigate to the folder where the files are downloaded and install pytest and pytest-cov using the below commands.

pip install pytest

pip install Pytest-cov

* To run the BlackboxTests.py run the below command

py.test BlackboxTests.py

* To run the WhiteboxTests.py run the below command

pytest --cov-report html --cov-branch --cov=. WhiteboxTests.py

* To see the branch coverage data, generate by the pytest-cov open the file with name WhiteboxTests\_py.html in the htmlcov folder which is generated by pytest-cov.

**Reflection on Blackbox and Whitebox Testing Techniques:**

As mentioned, both Blackbox testing and White-box testing have trade-offs.

To write the test suit for Blackbox, one has to have clear understanding of the requirement of the system. And each test case should be traceable with the requirement. Unlike White-box testing, this technique doesn’t need any prior knowledge of programming language as it is used to test the system from user’s perspective. However, the partitioning of the inputs plays an important role.

When it comes to White-box testing, the test cases are mainly focused around the internal program structure and hence to achieve maximum code coverage. It Is one of the better ways to identify the error when compared with Blackbox.

In my case, I had achieved 100% code coverage in both the testing techniques. For Blackbox I had to write six test cases to satisfy the equivalence partitioning of input space. Whereas for White-box I have achieved it with just three test cases. I don’t think by focusing solely on achieving coverage I can test the functional behavior. For instance, one testcase in my BlackboxTest.py named test\_for\_array\_with\_equal\_elements test the important functionality of binary search for array having same elements and I don’t have this test in my WhiteboxTests.py. however, I achieved 100% code coverage.