# Case Study Analysis and Impressions

## Infrastructure and tools-

1. 64 Bit Linux OS.
2. Pycharm IDE for unit test creation and execution.

## Files Used -

1. library.so file for execution.
2. hash.h file for expected behaviour reference.
3. Wrapper.py – The python equivalent of hash.h.

Deliverables -

1. Modified Wrapper.py to include all the return codes with reference to hash.h.
2. UnitTests.py, a single file testing the unit tests.

## Instructions to run the test-

1. Save the Auto\_All\_Tests.py under any name as preferred.
2. Run the test in an IDE of preference.
3. The test run sequentially with hashInit being called before every test and hashTerminate after every test.
4. The sample test results run at my personal environment with Directory = Source is as illustrated.

## Inconsistencies and Bugs-

1. Library loading issues and incompatibility issue-

* If the shared library (libhash.so) path is not given correctly, an error is seen. To avoid this, the correct path needs to be given

***OSError: libhash.so: cannot open shared object file: No such file or directory***

* If the library is built for different platform or architecture, the loading becomes a challenge. It is highly recommended to use the specific architecture. This issue can be overcome by creating Docker image and running the dependency in an isolated container.

1. HashDirectory–
   * Passing an invalid file path as an argument the function should fail, but it is executing successfully and it is logging predefined values in the log file. This case is handled as a unit test case but commented.

***HashReadNextLogLine: b'1 /initrd.img.old 0CFE7BAB96589E127A26082BEB3DAC'.***

***HashReadNextLogLine: b'1 /initrd.img 2032F928DFF8FE547DCE063F91146'.***

* + Passing a specific folder outside the projects source path should execute successfully, but this is failing and returning predefined values in log file.
  + Passing a sub folder under project root folder should execute successfully, but this is failing by hashing only the files under the current directory and ignoring the other files.
  + If a directory is passed as an argument, the total number of files which are successfully hashed after the execution of the function is not clear. It is also not clear what entry should be logged in case of a failure for correlation,
  + If a null value is passed as the directory path, the error is as shown below. This is not handled well.

***No Such File or Directory***

1. HashReadNextLogLine –

* If a source directory contains no files, the function logs two values in the log file as discussed earlier. The empty log case cannot be tested, and it is always assumed that the log file is populated.
* As HashDirecory is asynchronous, once initiated the HashReadNextLogLine should work, the log entry can be done at any time. During the execution of code, the HashReadLogLine code waits for the HashDirectory to complete execution and hangs in the middle.

1. HashStop and HashStatus– These functions cannot be tested to their full potential as it is assumed that the operation id is always valid. If an invalid operation id is given, the case is not handled as an exception.