**Assignment 2:Coverage Guided Fuzzing**

This assignment was about implementing two classes for a fuzzer which is about coverage guided fuzzing. I had to implement two methods for the First class and One method for the second class.

Following are the briefs about implementation of the methods:

**compareCoverage(curr\_metric, total\_metric):**

**It gets two arguments,using the curr\_metric which is coverage by the current testcase.I had looped through for each of the blocks present in curr\_metric and if any one is not present in total\_metric then I return True then and there else I test till all elements of curr\_metric is tested and none is new then I return False.**

**updateTotalCoverage(curr\_metric, total\_metric):**

**For this function the idea is we need to update the total\_metric of coverage info so that it keeps track of all coverages seen so far.So,I search for elements in curr\_metric and checks if its not present in total\_metric then append it.**

**mutate(input\_data,coverage\_info,irlist):**

**This is one of the most important element for a fuzzer having a good mutate function lets to achieve the best coverage for the given code in less time.**

**For the mutate function I had implemented it using combination of some individual methods (heuristics).So below are the details:**

1. **Addsub():This adds a random number between -100 to 100 to the input and returns.**
2. **Flip\_bit():This is like changes or negate one bit of the input bitwise and return.**
3. **Add\_bit():This adds a bit at some random place for the given input.**
4. **Remove\_bit():This removes a bit from any random position for the given input and returns.**

**Limitations:**

**The seed inputs matters the tool doesn’t give full cover for any seed input if the time is less hence the time limit either is sufficient or else the seeds needed to be good for the tool to give full coverage.**