CS6888 Program Analysis and its Applications
Assignment #2: Due October 8th 2PM

Assignment goal: to obtain hands-on experience on symbolic execution and its application to test generation in order to better understand the benefits, limitations, and tradeoffs of this kind of analysis approach.

This is an individual assignment. You are **not to share** your setup, benchmark development strategy, your results and analysis, or any other aspect of the assignment with your classmates.

Task 1: Setup Klee, a symbolic execution engine for test generation.

- We recommend using the docker installation at: https://klee.github.io/docker/
  - Downloading the image can take several minutes and about 1.5GB compressed
- Perform basic tutorials to get an idea of how to run Klee and its basic capabilities
  - https://klee.github.io/tutorials/testing-function/
  - https://klee.github.io/tutorials/testing-regex/
  - https://klee.github.io/tutorials/using-symbolic/
  - https://feliam.wordpress.com/2010/10/07/the-symbolic-maze/
- Scan following docs for space of options to consider and understand reported coverage
  - https://klee.github.io/docs/options/
  - http://ccadar.blogspot.com/2020/07/measuring-coverage-achieved-by-symbolic.html
  - https://klee.github.io/docs/tools/#klee-stats

Task 2: Develop benchmark of programs

Based on your understanding of symbolic and concolic execution and of Klee (as per its documentation and tutorials) you are to build a benchmark of small programs that showcases Klee's strengths and weaknesses. More specifically, you are to develop:

- 5 programs with diverse features that KLEE can handle to explore all paths
- 5 programs with diverse features that KLEE struggles with to explore all paths

The programs in the benchmark must:

1) Be developed **by you and only you** for this assignment
2) Be **distinct** from the ones included in the Klee demos and tutorials or papers using Klee
3) Be smaller than **50 lines of code** (when you run "`wc -l <file>`" on it),
4) Capture a **single specific program feature**. The target program feature must be used in the name of the program file. For example, if your program is meant to showcase how Klee deals with an iteration through a loop, you could call the program "loop-iteration.c".
5) Be **diverse**. You want each program to show a unique strength or a weakness in Klee that is not shown by the other programs in the benchmark. So, for example, showing how Klee handles a pointer in a program could be interesting, but also showing a program with

multiple pointers would not add much in terms of diversity. Similarly, have a program with the trigonometric function 'sin' could be interesting, but having another one with 'tan' would not add much in terms of showcasing what Klee can and cannot handle. You must aim to have a suite that explores different aspects of the language to show that diversity.

6) Be representative of program features used in practice. We are not looking for corner cases in a programming language that are not handled by Klee, but rather common coding constructs that we expect for it to handle.

Task 3: Write Report

The report must be **under 3 pages in pdf format**, and include:

- Program benchmark design considerations. For each of the 10 programs, explain the program feature considered and justify why it was selected in terms of diversity and representativeness.
- Create a table showing the number of tests and branch coverage achieved by Klee for each program in the benchmark (each program is a row in the table, the columns are program, # tests, and coverage). Make sure that each program clearly maps to one of the selected features discussed in the previous item.
- Discuss your findings.
  - Highlight and explain key observations revealed by your benchmark in terms of programming constructs that Klee handles either particularly well or badly.
  - Were you surprised by Klee's handling or mishandling of any construct? Explain why or why not.
  - For the constructs that are not handled well, conjecture why that is the case based on your understanding of symbolic execution.

Submission:

Submit through Collab under Assignment #2:

- Your benchmark source programs and a README with instructions to run each program in the benchmark
- A pdf version of the report

The grade will be based on the submitted benchmark and report, and the insights shown in their explanation. This assignment is worth 10% of your grade.