ECS 260 Software Engineering

Winter 2023

Homework #3

Due 5:00pm Thursday March 9th, 2023

Student Name: Yi-Hsiang Chiu, Yu-Cheng Hwang

1 Writing an LLVM Pass (70 pts)

- In this homework you will implement an LLVM pass that instruments any given program so that array out-of-bound accesses are reported dynamically. For example, if you have an array `int numbers[3]`, and your program attempts to access `numbers[3]` (or any other index outside the valid range of the array) at runtime, then an out-of-bounds access would be detected and reported:

```
1 #include <stdio.h>
2
3 int main() {
4   int numbers[3];
5   for(int i = 0; i < 4; i++) {
6     numbers[i] = i;
7   }
8   return 0;
9 }
```

  Your LLVM pass should print:

  tests/test1.c:6: Array Out of Bounds Error.

  Note that the source information (file name and line number) given as part of the error message will be retrieved by your LLVM pass. You cannot assume the program will always be called `test1.c`, for example.

- Your solution should be able to handle multi-dimensional arrays.

- This homework can be worked on individually, or in teams of **2 people**. Even when working as part of a team, **each team member is responsible for submitting the homework by the due date**. You will receive no credit in this assigment if you do not submit your own copy of the homework solution. Please include the name of your teammate in your source files as a comment.

- **IMPORTANT: Please start this homework early! Note that no partial credit will be given for code that does not compile or run. This part of the assignment will be graded based on the correctness of your implementation**

when run on the instructor's test programs. Thus, it is a good idea to test your solution as thoroughly as possible (see part 2 of the assignment).

- For this assignment, we will use a Docker image for LLVM 3.8. Visit the following link for instructions on how to install Docker: https://docs.docker.com/get-docker/

- Pull the Docker image:

  ```
  docker pull ucdavisplse/ecs260-hw3
  ```

- Create and run the Docker container:

  ```
  docker run -ti --ulimit="stack=-1:-1" --name hw3-llvm ucdavisplse/ecs260-hw3
  ```

- To start the Docker container after exit:

  ```
  docker start -ai hw3-llvm
  ```

  Warning!!! Backup your files *before* you exit the container, i.e., keep a copy of your files outside the Docker container. If your container gets killed, then you will *not* be able to recover your work. No homework extensions will be given if you accidentally lose your work.

- Download `handout.zip` from the Canvas, then copy it from your local file system to the container by executing the following command (this should be run outside of the docker container):

  ```
  docker cp <local-src-path> hw3-llvm:<dest-path>
  ```

  For example, the command can be: `docker cp ./handout.zip hw3-llvm:/home/llvm`

- Once inside the Docker container, unzip `handout.zip` and change to directory `handout`, which provides all necessary files and scripts to implement your solution. When you complete the assignment (i) zip `handout`, and (ii) submit the zip electronically via Canvas by the due date. Do not change the names of the files, the scripts, or the structure of the directory.

  Similarly, you can copy a file from the container to the local file system by executing the following command (this should also be run outside of the docker container):

  ```
  docker cp hw3-llvm:<src-path> <local-dest-path>
  ```

- The `handout` directory contains the following files:

  - `Instrument.hpp` and `Instrument.cpp`: the files where you will add your LLVM code to instrument the program. Comments have been included to guide you through the key parts of the solution.
  - `check_bounds.c`: the file where you will provide the implementation of your bound checking function.
  - `tests`: a directory where you will place your tests. There is an initial `test1.c`. Please follow same name convention for additional files.
  - `clean.sh`: a shell script to delete temporary files.

  - `compile.sh`: a shell script that compiles the LLVM Instrument files.
  - `run-tests.sh`: a shell script that runs the LLVM pass on all programs in the `tests` directory.
  - `expected.out`: the expected output when running `run-tests.sh`. Please update this file when adding more tests.

- You can also find in the Docker container the `sample-code` directory with the sample LLVM passes discussed in lecture. The pass "instrument" will be particularly useful when working on the homework.

- As you become familiar with the LLVM instruction set, it will be useful to look at the text representation of the instructions for `test1.c`.

  ```
  ./run-tests.sh
  cd tests
  llvm-dis test1.bc
  ```

  The above will produce `test1.ll`, which you can open with your favorite text editor.

- For LLVM documentation, please download the tar file from https://releases.llvm.org/3.8.0/llvm_doxygen-3.8.0.tar.xz. Open `index.html` in your browser.

- For general information about the LLVM language, you can refer to https://llvm.org/docs/LangRef.html.

2 Testing your LLVM Pass (10 pts)

- For this part of the homework, you will test your LLVM pass by adding additional tests to the `tests` directory. Please follow the file name convention: `testN.c` where N is the number of the test file. When done, please save the output you get from running the command `./run-tests.sh` in the file `expected.out`.

- This part of the assignment will be graded based on the completeness of your test suite, and will be graded independently of your implementation, thus it is possible to get full credit on this part even if your implementation does not give the expected results.

- Complete the table on Page 4 that lists the test case number, the test file name, a description of what it is testing and the expected output.

In summary, to complete the Homework #3 (i) zip `handout` (i.e., your code implementation including your tests), (ii) modify `hw3.tex` (i.e., filling the table on Page 4), (iii) create the corresponding PDF document, and (iv) submit both the zip and the separate PDF electronically via Canvas by the due date.

Test Case #	Test File Name	Description	Expected Output
1	tests/test1.c	1D array in a for loop	tests/test1.c:6: Array Out of Bounds Error.
2	tests/test2.c	2D array in a for loop	tests/test2.c:7: Array Out of Bounds Error.
3	tests/test3.c	1D array with start is -1	tests/test3.c:6: Array Out of Bounds Error.
4	tests/test4.c	2D array in a for loop	tests/test4.c:7: Array Out of Bounds Error.
5	tests/test5.c	3D array in a for loop	tests/test5.c:8: Array Out of Bounds Error.
6	tests/test6.c	1D double array in a for loop	
7	tests/test7.c	1D double array in a for loop	tests/test7.c:6: Array Out of Bounds Error.
8	tests/test8.c	2D double array in a for loop	tests/test8.c:7: Array Out of Bounds Error.