COMS W4115: Programming Assignment 4
Call Graph and Control Flow Graph

Logistics

1. **Announcement Date:** October 21st, 2019
2. **Due Date:** November 6^{th}, 2019 by 11:59pm. No extension!!
3. **Total Points:** 100

LLVM Pass for Call Graph

1. Run LLVM Call Graph pass

   (a) Run the following command to compile bubble.c to LLVM bytecode object bubble.bc.

   ```
   ./llvm-project/build/bin/clang -O0 -emit-llvm -c bubble.c
   ./llvm-project/build/bin/llvm-dis bubble.bc
   ```

   (b) Use the following command to run an example LLVM pass on bubble.bc.

   ```
   ./llvm-project/build/bin/opt  -dot-callgraph < bubble.bc
   dot -Tpdf ./callgraph.dot -o callgraph.pdf
   ```

2. Create an LLVM pass to generate call graph.

   In this assignment, you are required to create an LLVM pass to output a call graph dot file "callgraph.dot". In the llvm pass, you should use a module pass to analyze the caller-callee map and then output a dot file source code.
   Tips: In C++ you can write code such as outputting dot source code in constructors and destructors as well.

   Note: Reading and understanding the following source files may be helpful.
   "llvm-project/llvm/lib/Analysis/CallGraph.cpp",
   "llvm-project/llvm/lib/Analysis/CallPrinter.cpp",
   "llvm-project/llvm/include/llvm/Analysis/CallGraph.h",
   and "llvm-project/llvm/include/llvm/Analysis/CallPrinter.h".

   You can reference small code snippet in those files, but must not copy code directly from those files. Less than 100 LOC will be sufficient to solve this problem.

   Please rename your cpp file to {UNI}-callgraph.cpp and submit it. During grading, we will first rename your cpp file to hw4-callgraph.cpp and run it as follows.

   ```
   ./build/bin/opt  -load ./build/lib/LLVMcallgraph.so -hw4-callgraph < bubble.bc
   dot -Tpdf ./callgraph.dot -o callgraph.pdf
   ```

   Please define the created class name and registered LLVM pass name accordingly for full grade.

   We may also test your program on other C programs besides bubble.c.

LLVM Pass for Control Flow Graph

1. Run LLVM Control Flow Graph pass

   (a) Run the following command to compile bubble.c to LLVM bytecode object bubble.bc.

   ```
   ./llvm-project/build/bin/clang -O0 -emit-llvm -c bubble.c
   ./llvm-project/build/bin/llvm-dis bubble.bc
   ```

   (b) Use the following command to run an example LLVM pass on bubble.bc.

   ```
   ./llvm-project/build/bin/opt  -dot-cfg < bubble.bc
   ls -a
   dot -Tpdf .bubbleSort.dot -o bubblesort.pdf

   ./llvm-project/build/bin/opt  -dot-cfg-only < bubble.bc
   ls -a
   dot -Tpdf .bubbleSort.dot -o bubblesort.pdf
   ```

2. Create an LLVM pass to generate control flow graph

   In this assignment, you are required to create an LLVM pass to output a control flow graph dot file for each function definition. Each dot file should be named as {function name}.dot. In the pass, you should use function pass to analyze the control flow for each function and then output a dot file source code. The graph should be same as what "-dot-cfg-only" generates.
   Tips: In C++ you can write code such as outputting dot source code in constructors and destructors as well.

   Note: Reading and understanding the following source files may be helpful.
   "llvm-project/llvm/lib/Analysis/CFG.cpp",
   "llvm-project/llvm/lib/Analysis/CFGPrinter.cpp",
   "llvm-project/llvm/include/llvm/Analysis/CFG.h",
   and "llvm-project/llvm/include/llvm/Analysis/CFGPrinter.h".

   You can reference small code snippet in those files, but must not copy code directly from those files. Less than 150 LOC will be sufficient to solve this problem.

   Please rename your cpp file to {UNI}-cfg.cpp and submit it. During grading, we will first rename your cpp file to hw4-cfg.cpp and run it as follows.

   ```
   build/bin/opt -load ./build/lib/LLVMcfg.so -hw4-cfg < bubble.bc
   dot -Tpdf bubbleSort.dot -o bubblesort2.pdf
   ```

   Please define the created class name and registered LLVM pass name accordingly for full grade.

   We may also test your program on other C programs besides bubble.c.

Submission Guide

You can either work in pair or work by yourself.
Please submit the followings:

1. You are required to submit **{UNI}-callgraph.cpp**(50 points) and **{UNI}-cfg.cpp**(50 points). {UNI} means your UNI number.
2. Submit an extra file **contribution.txt** describing each of your contribution.