**Programming Assignment 2**

***Solution Description:***

My solution analyzes and instruments a program to path profile all inner loops. My solution consists of finding the innermost loop of a function, applying the “efficient path profiling” (EPP) algorithm, instrumenting the code accordingly, and printing the paths. When the newly instrumented program is run, the paths that are taken are counted and outputted to the screen.

Here is a more in depth explanation of my solution through pseudocode:

* If loop contains no other loops
  + Traverse basic block graph in reverse topological order
    - For each block visited
      * Calculate and store the number of paths and edge values per basic block and corresponding edges according to EPP algorithm in figure 5
      * If edge is exiting loop, place dummy node *Exit* for place holder
  + Insert instrumentation into code
    - Create *Initialize Path Register, Increment Path Register,* and *Finalize Path Register* functions
    - Insert *Initialize* function at header basic block
    - Iterate through all basic blocks to insert *Increment Path Register*
      * If *Exit* (except *latch)*, insert *Increment* and *Finalize* function at actual *Exit* block
      * Else, if no value, then insert *Increment* at appropriate basic block
    - Insert *Finalize* function for latch
  + Print path
    - Iterate through using DFS to find and print all paths
* Repeat on next loop

***Structure of Archive:***

* *InstrumentPass.cpp*: contains the solution
* *epp\_runtime.cpp:* contains the runtime library definitions
* */test*: folder contains all test code
* *\*.jpeg*: files show the CFG of the desired instrumented code
* *path-desc.ll* and *path-pro.ll* contain the outputs for loop ID, path ID, path, and number of iterations

***Script Changes and Code Testing:***

*run.sh* file changes:

* inserted *-dot-cfg* option in line 10 to output CFG representation *.dot* file
* inserted $2 instead of 10 in line 16 to allow for user input for usually the number of loop iterations
* uncommented line 11 and inserted *IRPrint.txt* to output code information into that file
* inserted *dot -Tjpeg cfg.main.dot -o $1.jpeg* to create a jpeg file of the CFG graph for debugging

*Makefile* changes: Added command to remove .*jpeg* and *.dot* files when cleaning

To test the code, simply enter *./run.sh <.cc file name to instrument> <number of loop iterations>*

Example: *./run.sh simpleloop 10*