Exercise 2:

Write a Pintool (in JIT mode) that prints into a file called "**loop-count.csv**" the profiling about executed loops in each routine (RTN).

No need to handle loops that are implemented using indirect jumps.

The pintool should be named "**ex2.so"**.

For each loop with a non-zero **CountSeen**, the tool should emit the following information, in this exact format:

0x <loop$_1$ target address>, <loop$_1$ CountSeen>, <loop$_1$ CountLoopInvoked>, <loop$_1$ MeanTaken>, <loop$_1$ DiffCount>, <loop$_1$ routine name>, 0x <loop$_1$ routine address> , < instructions count of RTN containing loop$_1$>, <number times RTN$_1$ was called>

0x <loop$_2$ target address>, <loop$_2$ CountSeen>, <loop$_2$ CountLoopInvoked>, <loop$_2$ MeanTaken>, <loop$_2$ DiffCount>, <loop$_2$ routine name>, 0x <loop$_2$ routine address> , < instructions count of RTN containing loop$_2$ >, <number times RTN$_2$ was called>

…

0x <loop$_n$ target address>, <loop$_n$ CountSeen>, <loop$_n$ CountLoopInvoked>, <loop$_n$ MeanTaken>, <loop$_n$ DiffCount>, <loop$_n$ routine name>, 0x <loop$_n$ routine address> , < instructions count of RTN containing loop$_n$>,

<number times RTN$_n$ was called>

Where:

CountSeen = total number of times the loop's backward edge was executed

CountLoopInvoked = number of times the loop was invoked

MeanTaken = average number of iterations taken for the loop invocations

DiffCount = number of times that two successive loop invocations took a different number of iterations

routine name/address/ = Routine name/address in which the loop resides and the number of times it was called.

routine exec count = instructions count of the routine containing the loop (see exercise 1)

The above loops' list should be ordered according to highest **CountSeen** down to lowest **CountSeen**.

You can assume that the total number of loops is no larger than 10,000 and number of total routines no larger than 1000.

The pintool should not run longer than 1 second (elapsed time) on the bzip2 input.

Tips:

Cnsider using the API ***INS_DirectControlFlowTargetAddress(ins)*** to retrieve the direct target address and compare it to ***INS_Address(ins)***

*See **jumpmix.cpp*** on how to collect statistics on taken vs. non-taken conditional branches.

Test your pintool:

In the moodle you'll find the input binary file called "**bzip2.gz**" along with an input file to give it called "**input.txt.gz.**

Ftp the files to your T2 Linux account and open them using the **gunzip** command.
To run it simply type: $ **./bzip2 –k –f input.txt**
This will compress the file **input.txt** and generate a new file **input.txt.bz2**

To test your pintool on the above **bzip2** binary file, simply type:
<pindir>/pin –t ex2.so -- ./bzip2 –k –f input.txt

Submission requirements:
The submission of this exercise is **in pairs only**.
Submit 1 compressed file called **"ex2.zip"** into the moodle exercise2 link containing the following files:

1. The binary of your pintool **ex2.so** (compiled, and tested by you that it runs and gives the result).
2. A directory called: '**src**' containing all the **source files** (.cpp and .h files) of your pintool along with the "**makefile**", "**makefile.rules",** and a **REDAME.txt** file that includes your **full name**, your **ID** and a description of the compilation command and how to run the tool.

Submission deadline: extended to midnight Thursday, June 1st, 2023.