Exercise 4:

Write a Pintool in Probe mode named "**ex4.so**" that extends exercise 3.

Similar to exercise 2, for each basic block (bbl) with a non-zero execution count, the tool should emit the following information, in the following format:

<bbl$_1$ address>, <bbl$_1$ exec count>, <taken count>, <fallthru count >, <exec count of each indirect target jump up to 4 target addresses: <target addr$_1$, exec count>, <target addr$_2$, exec count>,…< target addr$_4$, exec count>>

<bbl$_2$ address>, <bbl$_2$ exec count>, <taken count >, <fallthru count>, <exec count of each indirect target jump up to 4 target addresses : <target addr$_1$,exec count>, <target addr$_2$, exec count>,…< target addr$_4$, exec count>>

…

<bbl$_n$ address>, <bbl$_n$ exec count>, <taken count >, <fallthru count>, <exec count of each indirect target jump up to 4 target addresses:: <target addr$_1$, exec count>, <target addr$_2$, exec count>,…< target addr$_4$, exec count>>

The basic blocks should be **sorted** from most frequently executed ("hottest") to the least frequently executed ("coldest") ones.

You can assume that the total number of basic blocks is less than 10,000.

For the exercise it is recommended to use the pintool **ex3.cpp** you created in exercise 3 which was based on **btranslate.cpp** located at:
https://moodle24.technion.ac.il/mod/resource/view.php?id=151032

Recommended tip:

Use the following XED code to extract the needed information from a given indirect jump instruction:
jmp <targ_reg> e.g., "*jmp rax*"

Or : "***jmp [<base_reg> + <disp> + <index_reg> * <scale>],*** e.g., "*jmp [rip+0x100+rax*8]*"

```
// Retrieve the details about the indirect jmp operands of 'INS ins':
xed_decoded_inst_t *xedd = INS_XedDec(ins);
xed_reg_enum_t base_reg = xed_decoded_inst_get_base_reg(xedd, 0);
xed_reg_enum_t index_reg = xed_decoded_inst_get_index_reg(xedd, 0);
xed_int64_t disp = xed_decoded_inst_get_memory_displacement(xedd, 0);
xed_uint_t scale = xed_decoded_inst_get_scale(xedd, 0);
xed_uint_t width = xed_decoded_inst_get_memory_displacement_width_bits(xedd,0);
unsigned mem_addr_width = xed_decoded_inst_get_memop_address_width(xedd,0);

xed_reg_enum_t targ_reg = XED_REG_INVALID;
unsigned memops = xed_decoded_inst_number_of_memory_operands(xedd);
if (!memops)
  targ_reg = xed_decoded_inst_get_reg(xedd, XED_OPERAND_REG0);

// Debug print.
dump_instr_from_xedd (xedd, ins_addr);
cerr << " base reg: " << xed_reg_enum_t2str(base_reg)
     << " index reg " << xed_reg_enum_t2str(index_reg)
     << " scale: " << scale
     << " disp: " << disp
```

```
<< " width: " << width
<< " mem addr width: " << mem_addr_width
<< " targ reg: " << targ_reg << xed_reg_enum_t2str(targ_reg)
<< "\n";
```

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

To test your pintool on the above **bzip2** binary file, simply type:
$ time <pindir>/pin –t ex4.so -- ./bzip2 –k –f input-long.txt

Your pintool should not run longer than 10 seconds (elapsed time) on the bzip2 input.

Tips:

Work in stages as follows:

Submission requirements:
The submission of this exercise is **in pairs only**.
Submit 1 compressed file called **"ex4.zip"** into the moodle exercise 3 link containing the following files:
1. The binary of your pintool **ex4.so** (compiled, and tested by you that it runs and gives the result).
2. A directory called: 'src' containing all the sources of your pintool along with the make files '**makefile'**, **makefile.rules**' and a **REDAME.txt** file that includes the following:
a. names + id numbers
b. How to run the tool.

Submission deadline is Thursday, July 10, 2025 at midnight.