Assignment 3

How to run

PIN VERSION - `pin-3.13`

1. Extract the folder and place `yatna_pintool` directory at PATH `/pin-3.13-98189-XXX/source/tools/` (same directory where `MyPinTool` is present)
2. Make using - `make all TARGET=ia32`
3. Run using - `./mypin  PIN_PROGRAM_NAME  TEST_EXECUTABLE_NAME  FLAGS (optional flags upto 5)`

IMPORTANT: If there are >5 flags being passed to the test program (eg `ls arg1 arg2 arg3 arg4 arg5 arg6`) , add corresponding $x in `mypin` script (line 2)
OR
Execute the 2 lines in `mypin` manually, output is present in `out_program_name` file

List of `PIN_PROGRAM_NAMES`

1.	`bb_count`		(Warmup part 1)
2.	`malloc_count`		(Warmup part 1)
3.	`cfi_count`	(bonus)	(Warmup part 1)
4.	`btrace`		(Security application 3.1)
5.	`stack` -	(bonus)	(Security application 3.2)

List of `TEST_EXECUTABLE_NAME`

1.	`./sample/test_hw`	(Sample Hello World executable)
2.	`./sample/test_malloc`	(Sample Hello World executable with malloc calls)
3.	`./sample/test_recur RECURSION_DEPTH`	(Sample recursive program executable)
4.	`/bin/ls`	
5.	`/bin/cat some_file_name`	
6.	`nano`	
7.	`vim`	
8.	`gedit`	
9.	`Libreoffice`	
10.	`Any Others`	

Warmup (all support multithreading)

1. Basic Block Count

   Run eg.
   ```
   ./mypin bb_count ./sample/test_hw
   ./mypin bb_count /bin/ls
   ./mypin bb_count gedit (takes ~2mins)
   ```

2. Malloc Count & Memory Allocated

   Run eg.
   ```
   ./mypin malloc_count ./sample/test_malloc
   ./mypin malloc_count /bin/ls
   ./mypin malloc_count gedit (takes ~1.5mins)
   ```

3. Control Flow Transfer Count (Bonus)

   Run eg.
   ```
   ./mypin cfi_count ./sample/test_malloc
   ./mypin cfi_count /bin/ls
   ./mypin cfi_count gedit (takes ~1.5mins)
   ```

Security Application

1. System Call Interception - **btrace** (supports multithreading)

Implemented around **25 system calls** in a similar print format to strace. Rest are shown as `some_unimplemented_system_call(...params...) = ret_value`

Run eg.
```
./mypin btrace ./sample/test_malloc
./mypin btrace /bin/ls
./mypin btrace gedit (takes ~2mins)
```

2. Stack Use Analysis & Stack Pivoting Detection (supports multithreading)

Aborts with error message in case of stack pivoting. Otherwise displays stack size of each thread.

Run eg.

```
./mypin stack ./sample/test_recur 11
./mypin stack ./sample/test_recur 12
./mypin stack ./sample/test_malloc
./mypin stack /bin/ls
./mypin stack gedit (takes ~2mins)
```