Prog 4: Cache Simulation

Develop a program that accepts the name of a memory trace file as a command line argument. Use the data to simulate a set associative cache using LRU replacement. Additional command line arguments will be needed to identify the details of the simulated cache. The arguments should appear in the following order:

1. X, where 2^X is the number of direct-mapped sets
2. Y, where 2^Y is the number of blocks per set
3. Z, where 2^Z is the number of addresses cached in each block
4. the memory trace input file

Your program (say it's a compiled C program) could be run like this:

```
./cache 3 2 4 data.tra
```

This would simulate an 8 set associative LRU cache with 4 blocks per set, where each block caches 16 addresses.

Trace File

The trace file is line based and in order with each line formatted:

- memory address requested, in hex
- tab
- 'R' or 'W' indicating the access was a read or write, respectively (unused)

Please note this data is real, so the addresses are larger than 32-bits. In languages like C/C++, you should store these values in pointers or uint64_t variables (made available through the headers stdint.h and cstdint for C and C++, respectively).

Program output

When finished analyzing the trace data, your program should print the miss ratio (the total number of missed accesses / the total access count). The output should be formatted EXACTLY as below with only the numbers varying. Please note the total number of memory accesses in my test trace may exceed the capacity of a signed 32-bit value, so choose your data types wisely. Example proper output would be (I made up the numbers):

```
missed 1234 / 234567
```