本次实验要求实现一个缓存模拟器，要求达到和 csim-ref 一样的效果。
首先我们需要准备一些结构体用于后续的操作。

结构体

line 结构体

line 结构体用于存储一行的信息，包括 isValid 表示有效位，tag 表示 tag 位，timestamp 为时间戳用于替换的 LRU 算法。

operation 结构体

operation 结构体用于存储一个操作，包括操作类型 type，操作的长度 size，以及地址 address

cache 结构体

cache 结构体用于存储一个缓存块，包括行数、组数、block 比特位数、set 比特位数、缓存块指针。

下面是需要的函数

函数

getSet 函数

用于获得 set 位，即 address 右移掉 block 的位，通过模 2 的 S 次方获得 set 位。

getTag 函数

返回 address 右移掉 block 位和 tag 位的结果即可。

allocateSet 函数

用于给缓存分配内存空间的函数，即初始化一块缓存。

freeSet 函数

和分配相对应，即完成所有操作后调用来释放缓存所占的内存空间。

readOneOperation 函数

从文件读取一行操作（即一个操作），将操作标准化后存入 operation 结构体并返回。

getArguments 函数

用于读取命令行参数的函数，采用 getopt 函数读取即可。

findMatchLine 函数

用于找到匹配 tag 的行，若匹配上则返回对应的行，反之返回 NULL。

findNewLine 函数

用于找到一个空闲的行，若找到了则返回这个行，否则返回 NULL。

findEvictLine 函数

用于找到要剔除的行，利用 LRU 算法，根据 timestamp 来寻找。

loadOrStore 函数

用于加载和存储操作的函数，即要找到对应的内存块判断是否命中。

modify 函数

分别用 Load 和 Store 来调用 LoadOrStore 函数

完成了准备工作，我们需要来完成主函数。
主函数中，我们首先对于命令行参数进行初始化，然后读入，随后打开 trace 文件来一行一行读取操作，完成后打印结果信息，释放内存。

完成代码后我们来测试一下

```
harry@ubuntu22:~/Code/CSAPP-LAB/cachelab/10225501403/cachelab-handout$ make
gcc -g -Wall -Werror -std=c99 -m64 -o csim csim.c cachelab.c -lm
harry@ubuntu22:~/Code/CSAPP-LAB/cachelab/10225501403/cachelab-handout$ ./test-csim
                        Your simulator     Reference simulator
Points (s,E,b)    Hits  Misses  Evicts    Hits  Misses  Evicts
     6 (1,1,1)       9       8       6       9       8       6  traces/yi2.trace
     6 (4,2,4)       4       5       2       4       5       2  traces/yi.trace
     6 (2,1,4)       2       3       1       2       3       1  traces/dave.trace
     6 (2,1,3)     167      71      67     167      71      67  traces/trans.trace
     6 (2,2,3)     201      37      29     201      37      29  traces/trans.trace
     6 (2,4,3)     212      26      10     212      26      10  traces/trans.trace
     6 (5,1,5)     231       7       0     231       7       0  traces/trans.trace
     8 (5,1,5)  265189   21775   21743  265189   21775   21743  traces/long.trace
    50

TEST_CSIM_RESULTS=50
```

可以发现成功实现了 csim-ref。