

Project4

Lock Table

Project Hierarchy

- You should submit your project in a directory structure like this:
“your_repo/project4/lock_table”
- Follow the lock_table directory structure as before.

```
ktlee20@multicore-36:~/TA/2021_DB/db2021/project4$ tree -a
.
├── .clang-format
├── CMakeLists.txt
├── .gitignore
├── lock_table
│   ├── CMakeLists.txt
│   ├── include
│   │   └── lock_table.h
│   └── src
│       └── lock_table.cc
└── main.cc

3 directories, 7 files
```



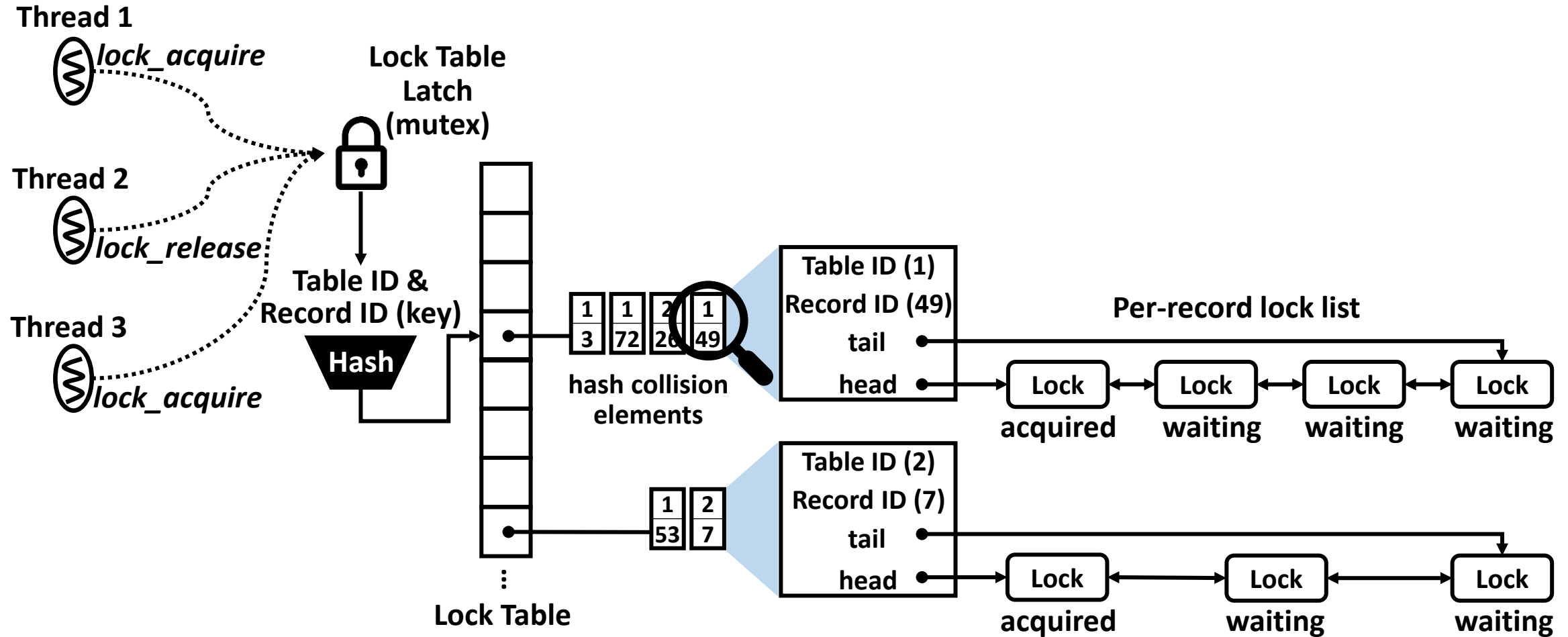
The output file must be an **executable file** named ***test_lock_table*** in ***bin directory***,

- ***not a library file as usual.***

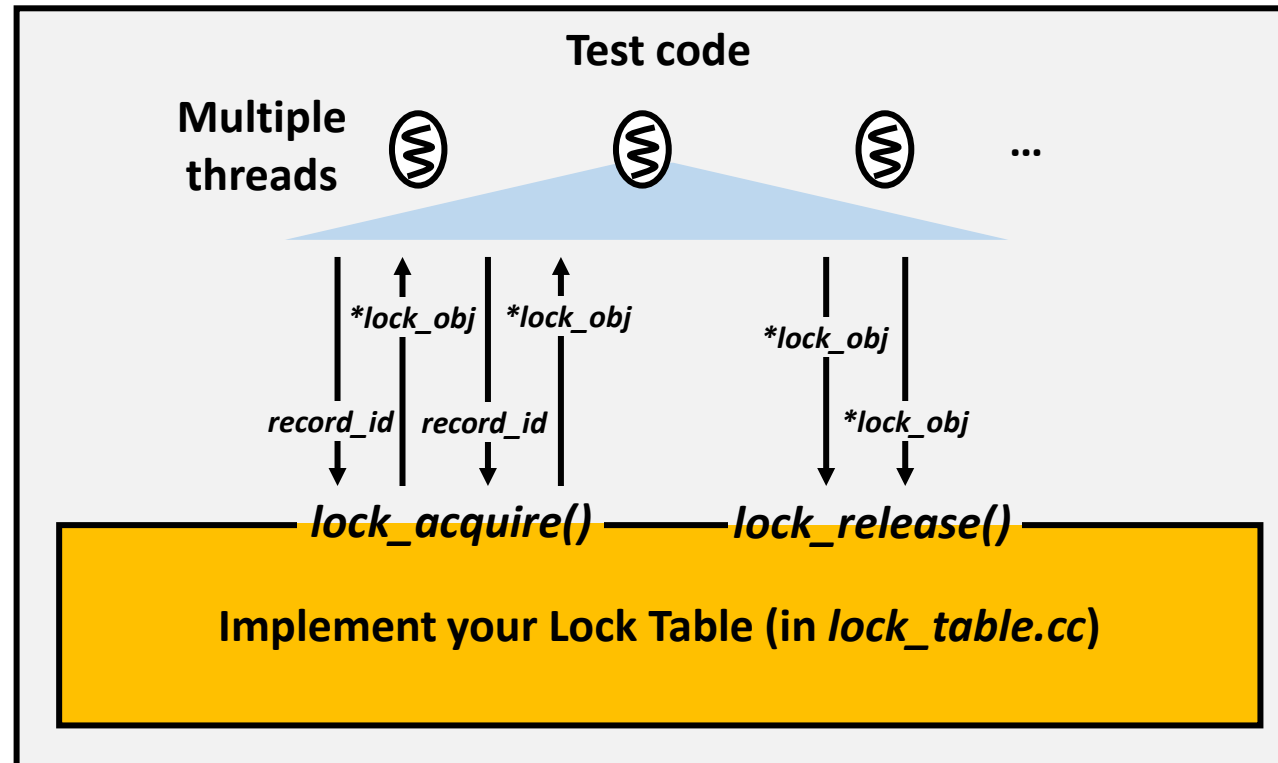
Project Overview

- Your task is to implement a **lock table** module that manages lock objects of multiple threads.
- The module doesn't need to be compatible with your developing database in this step.
- Instead, the module **should be correctly working with the given test code**.
- This project is a prerequisite step for the next project, Concurrency Control.
- Design your lock table and describe it on hconnect **Wiki** page.

Overall Architecture



Overall Architecture



Lock Table APIs

int init_lock_table(void)

- Initialize any data structures required for implementing lock table, such as hash table, lock table latch, etc.
- If success, return 0. Otherwise, return non-zero value.

lock_t lock_acquire(int64_t table_id, int64_t key)*

- Allocate and append a new lock object to the lock list of the record having the *key*.
 - If there is a predecessor's lock object in the lock list, **sleep** until the predecessor to release its lock.
 - If there is no predecessor's lock object, return the address of the new lock object.
- If an error occurs, return NULL.

int lock_release(lock_t lock_obj)*

- Remove the *lock_obj* from the lock list.
 - If there is a successor's lock waiting for the thread releasing the lock, **wake up** the successor.
- If success, return 0. Otherwise, return non-zero value.

Lock Table APIs

```
lock_t* lock_acquire(int64_t table_id, int64_t key)
{
    pthread_mutex_lock(&lock_table_latch);

    ...

    pthread_mutex_unlock(&lock_table_latch);
    return ...
}
```

```
int lock_release(lock_t* lock_obj)
{
    pthread_mutex_lock(&lock_table_latch);

    ...

    pthread_mutex_unlock(&lock_table_latch);
    return ...
}
```

Protect the acquire and release function as a **critical section** so that only one thread should be able to access the lock table at a time.

Hash Table

Combine *table id* and *record id (key)* in your own way, and use it as an input of the hash function

Table ID &
Record ID (key)

Hash

You can use any hash table (custom, stl, open-sourced, ...) that properly **resolve hash collision**, so that each <table id & record id> is uniquely distinguishable.

1	1	2	1
3	72	26	49

hash collision
elements

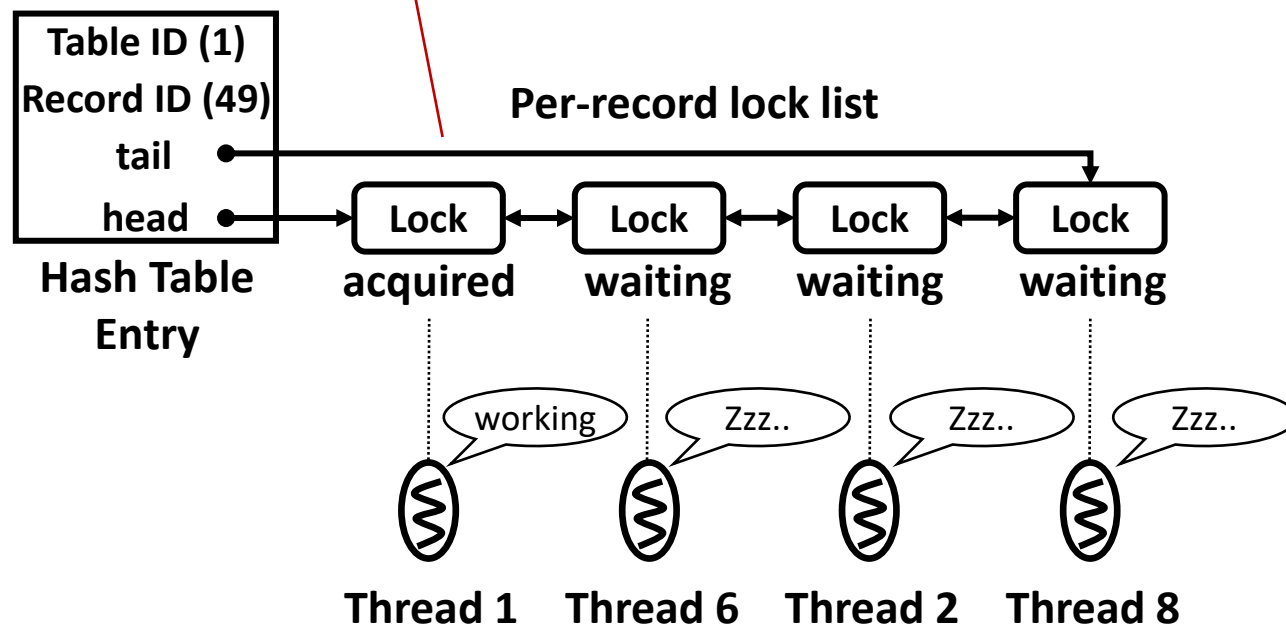
1	2
53	7

⋮
Lock Table

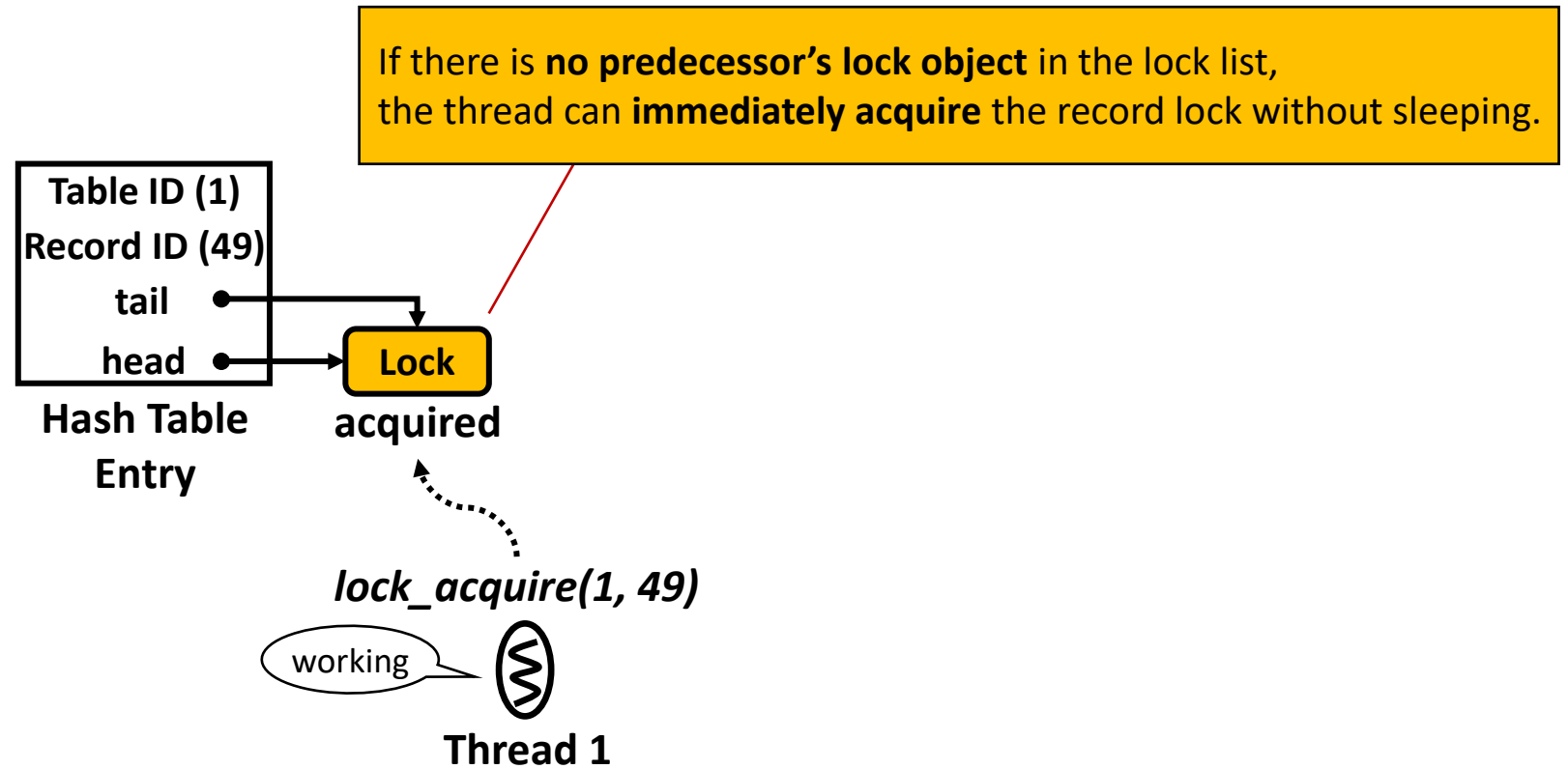
There is **no limitation of the size** of the hash table.

Lock List

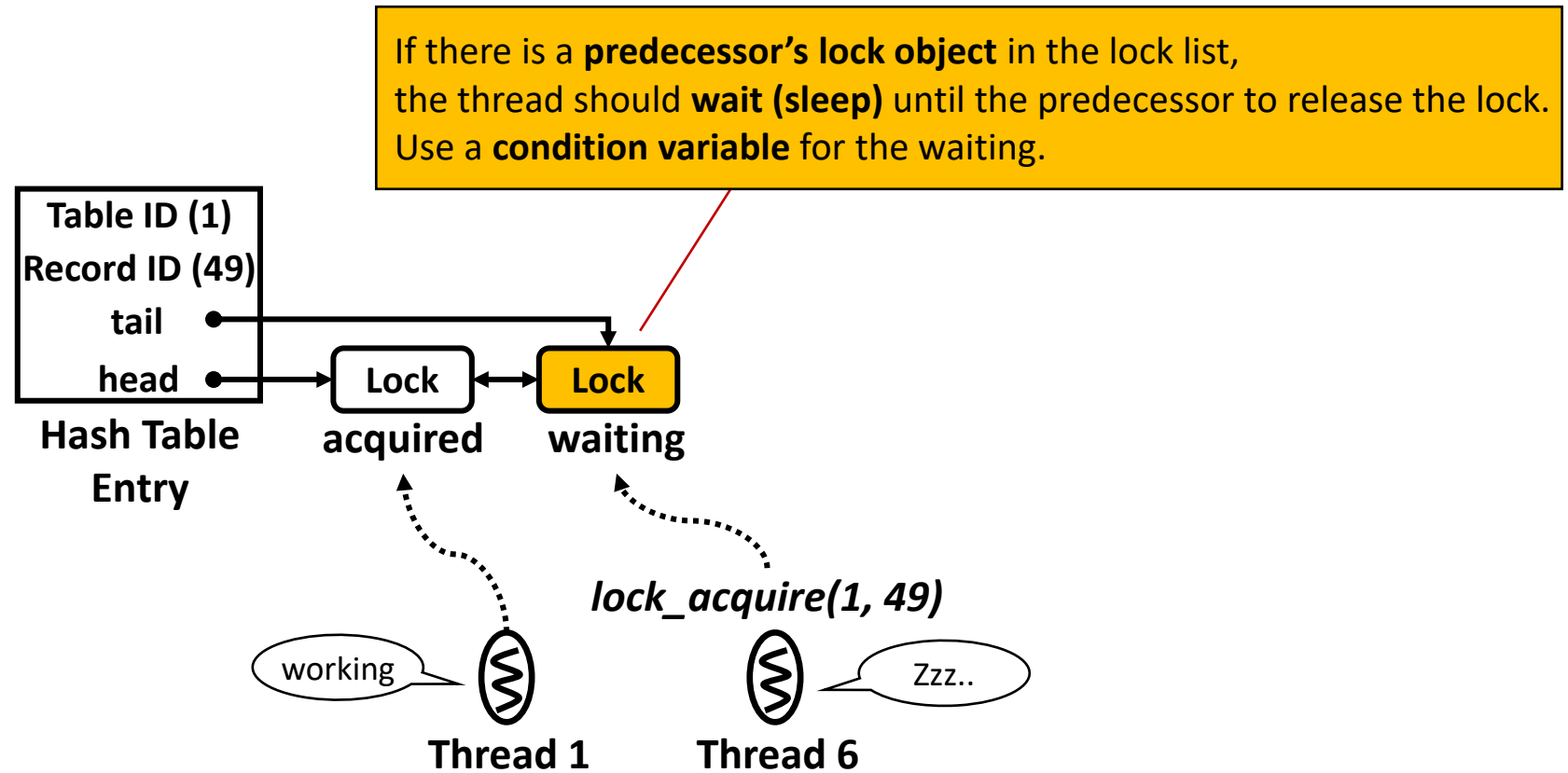
For each record, maintain a **lock list** that links lock objects of multiple threads.



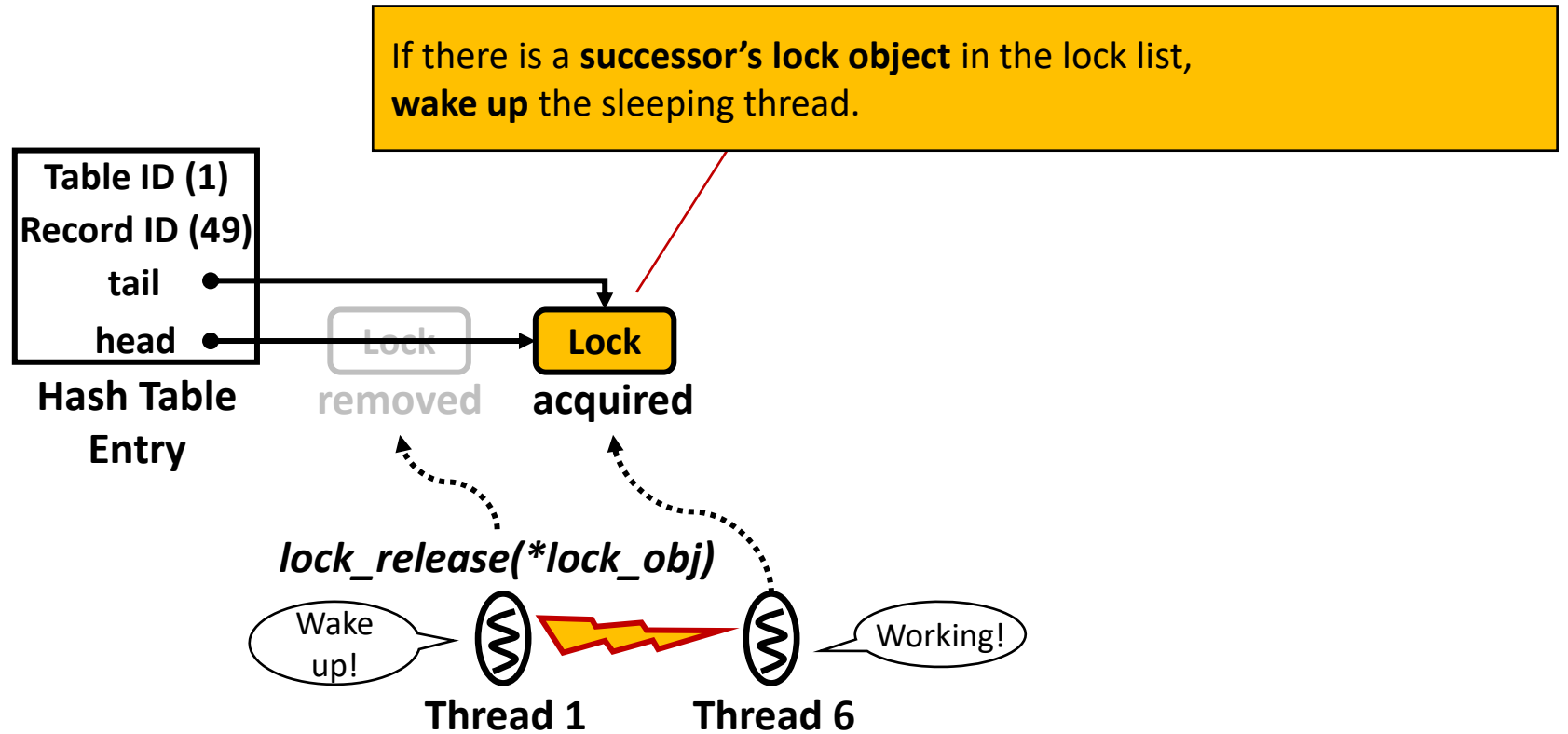
lock_acquire()



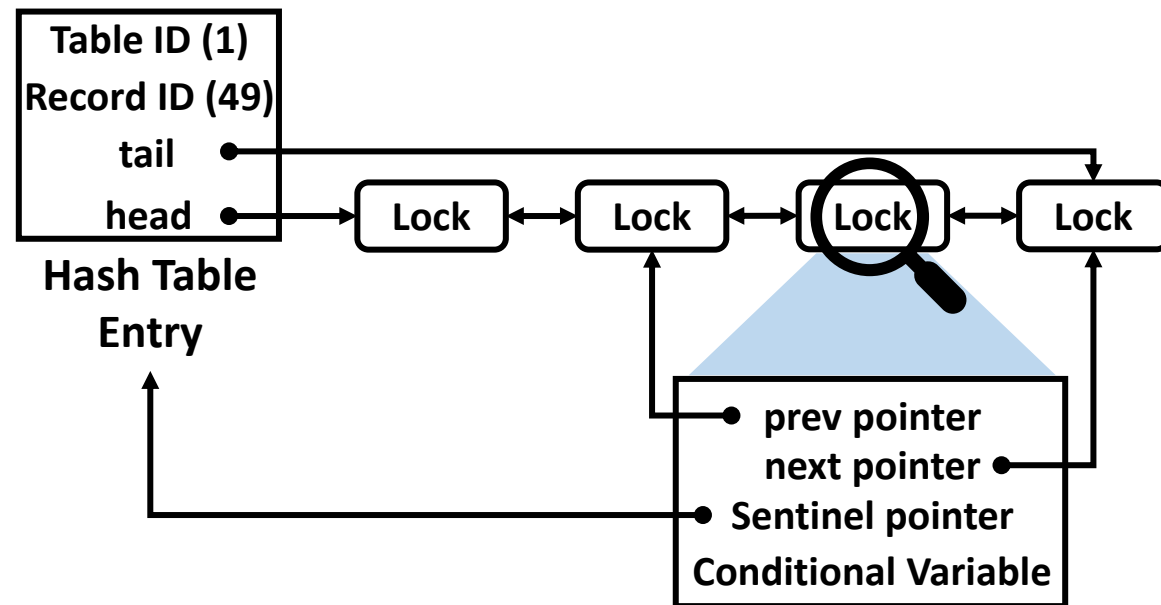
lock_acquire()



lock_release()



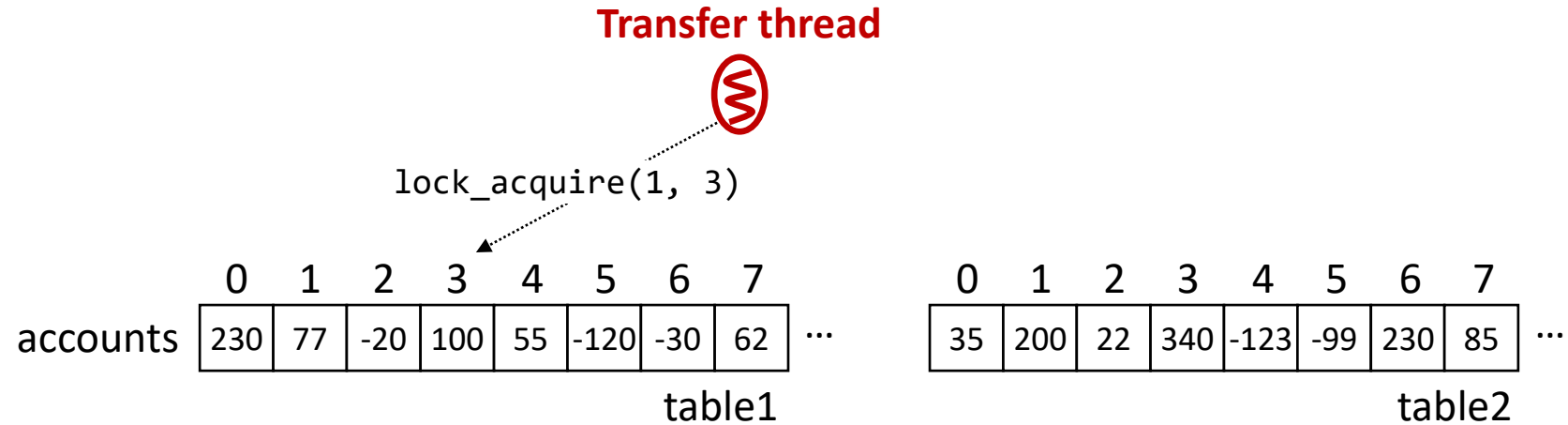
Lock Object



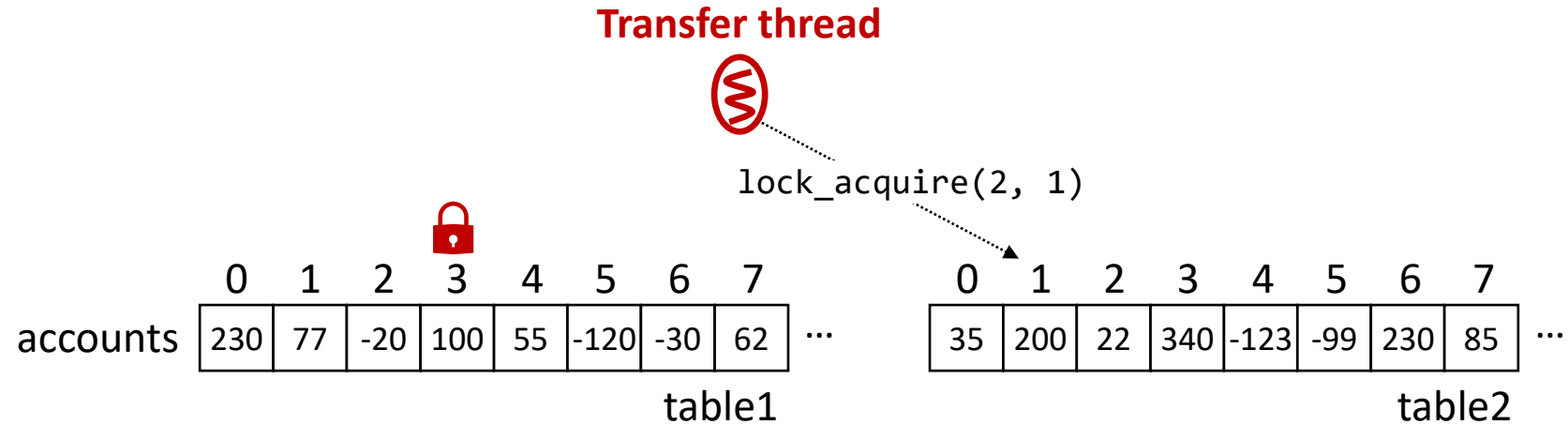
Test Code

- The given test code(main.cc) will
 - call *init_lock_table()* API function,
 - create multiple threads each of which
 - repeatedly acquire and release multiple record locks by calling *lock_acquire()*, *lock_release()*.
- The test code will safely schedule the operations avoiding deadlock, so you **don't have to deal with the deadlock problem** in this project.
- Analyze the test code as much as you want.

Test Code

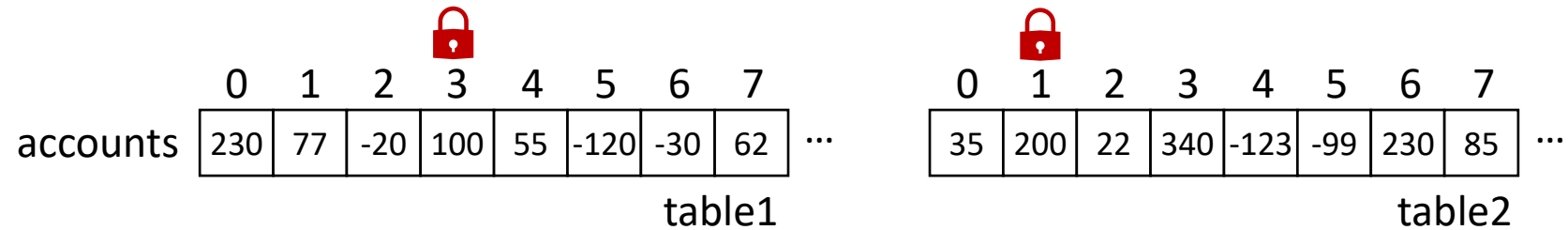


Test Code

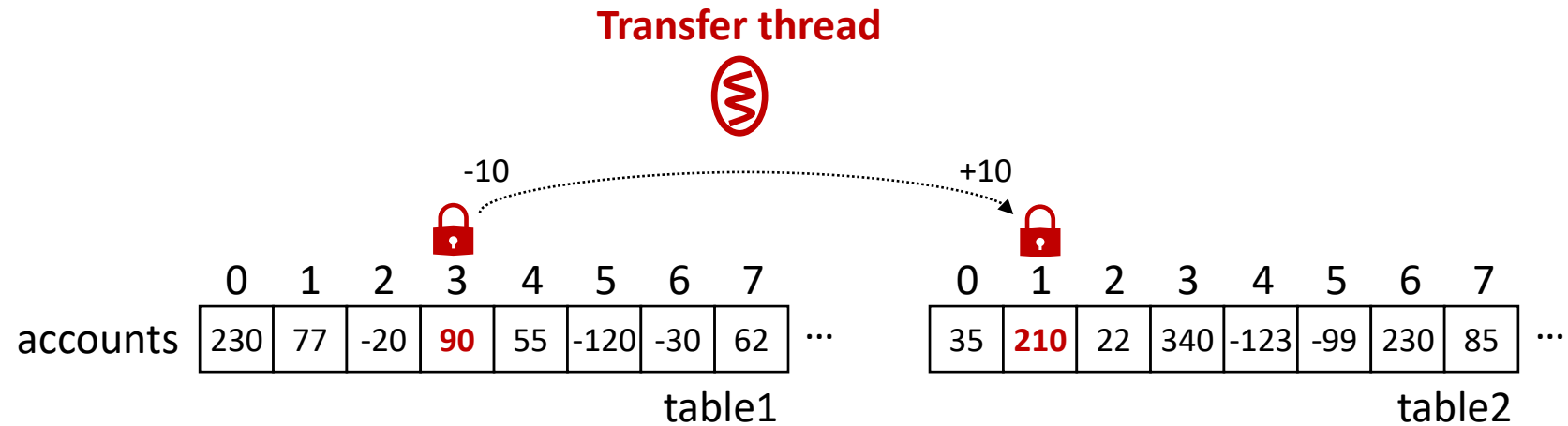


Test Code

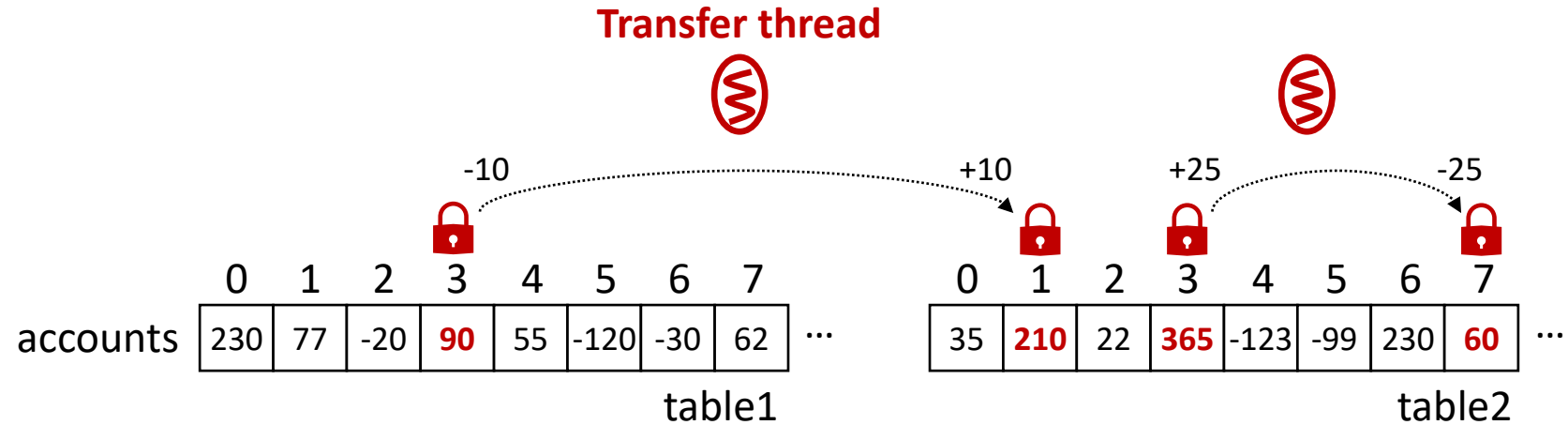
Transfer thread



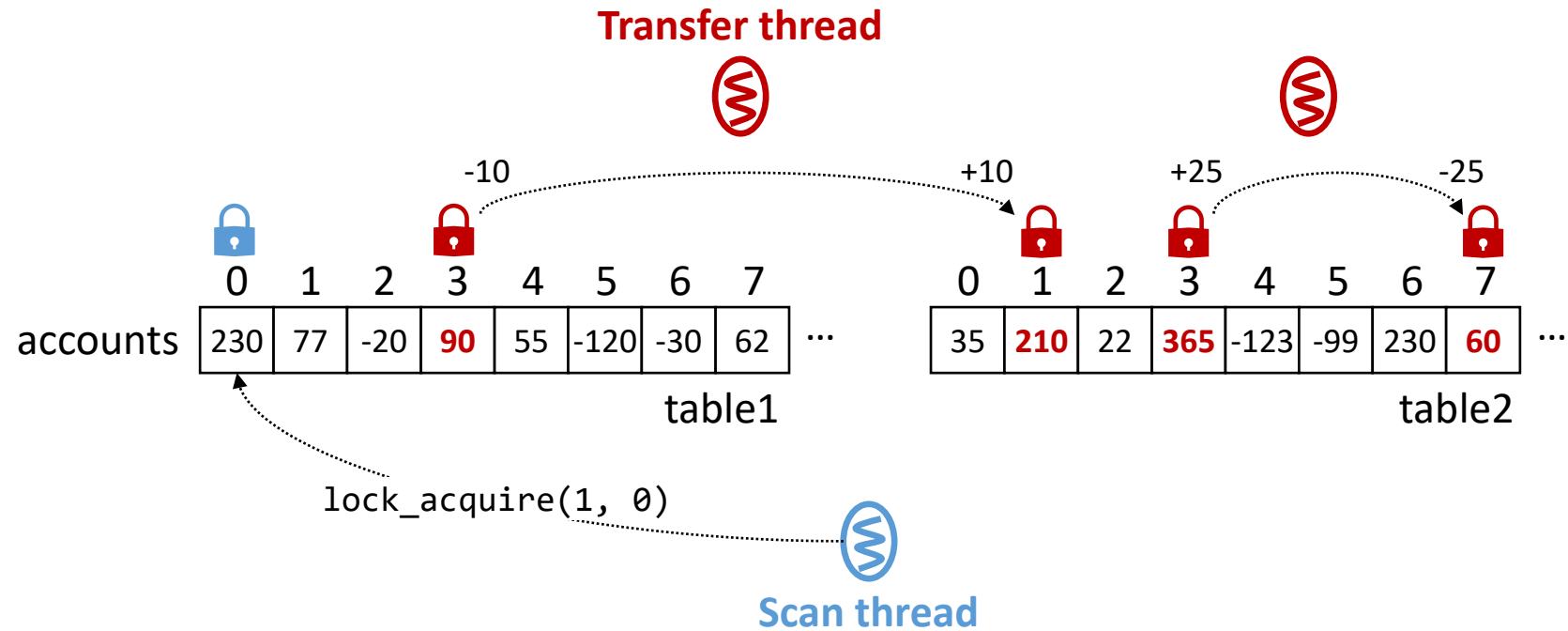
Test Code



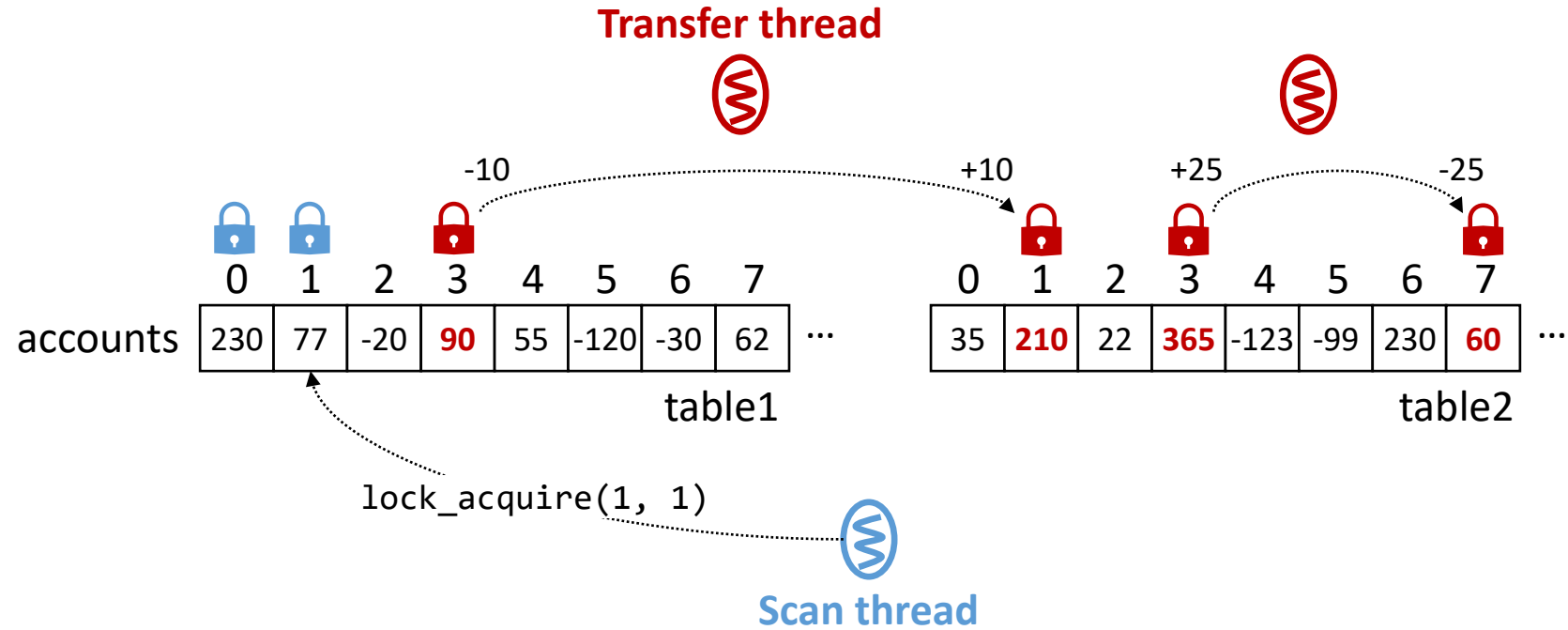
Test Code



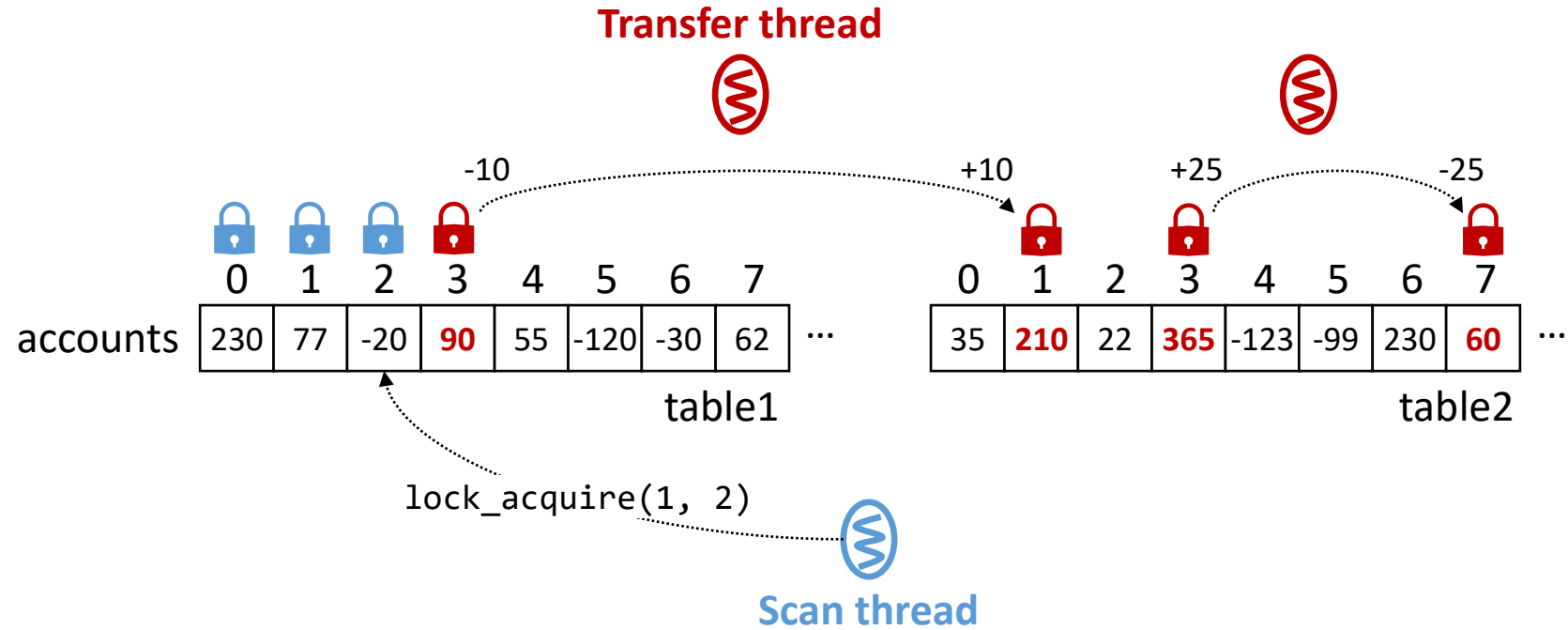
Test Code



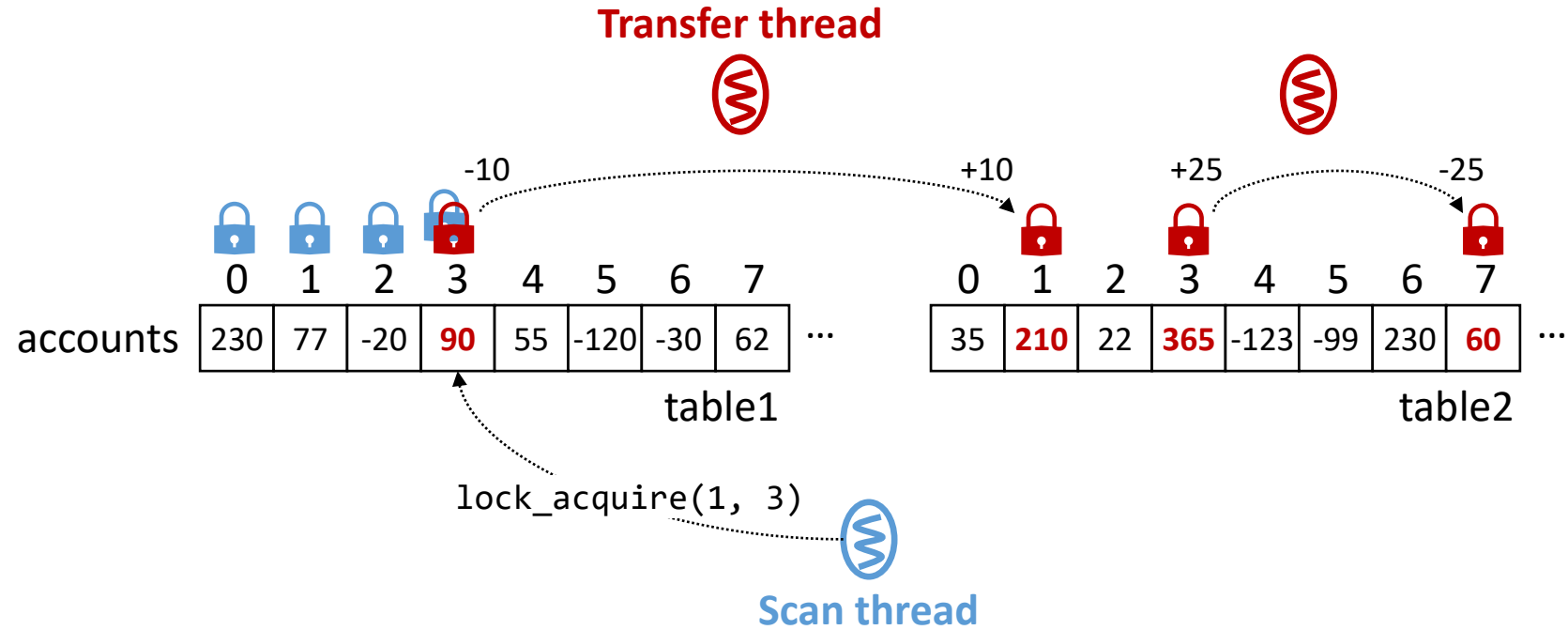
Test Code



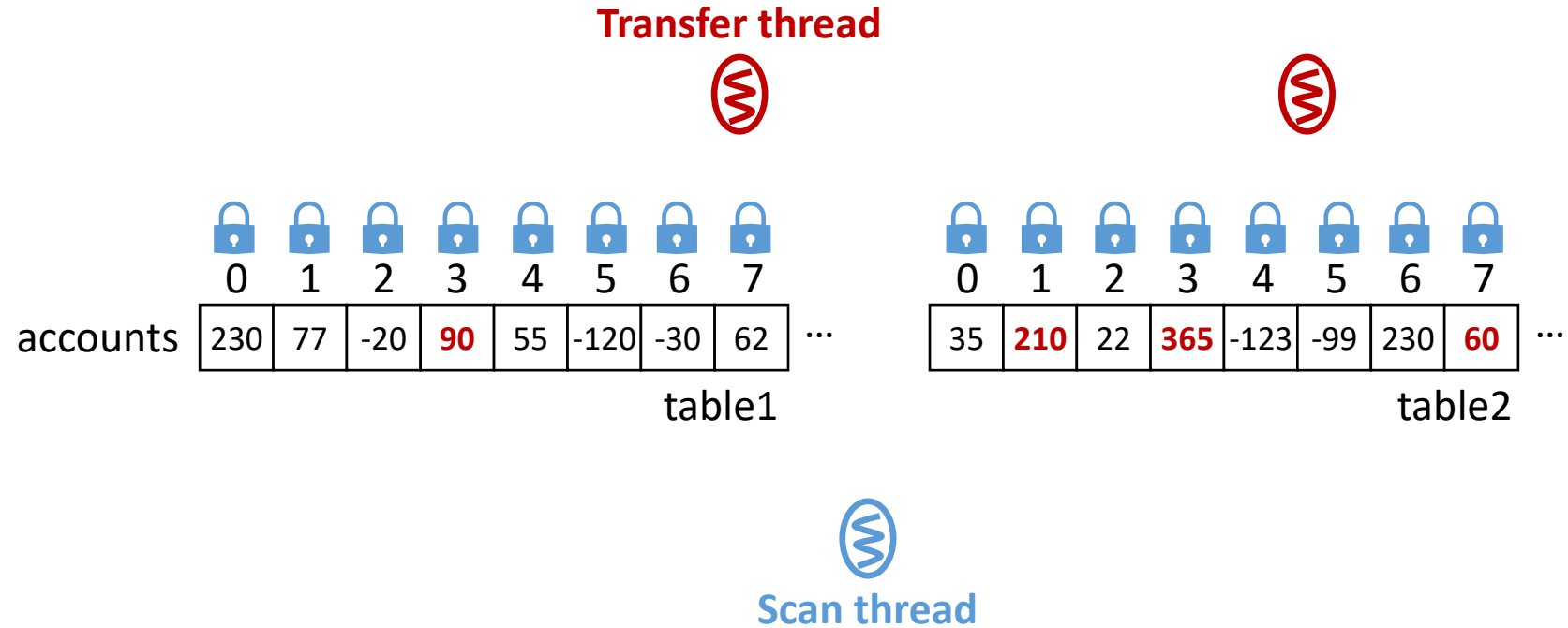
Test Code



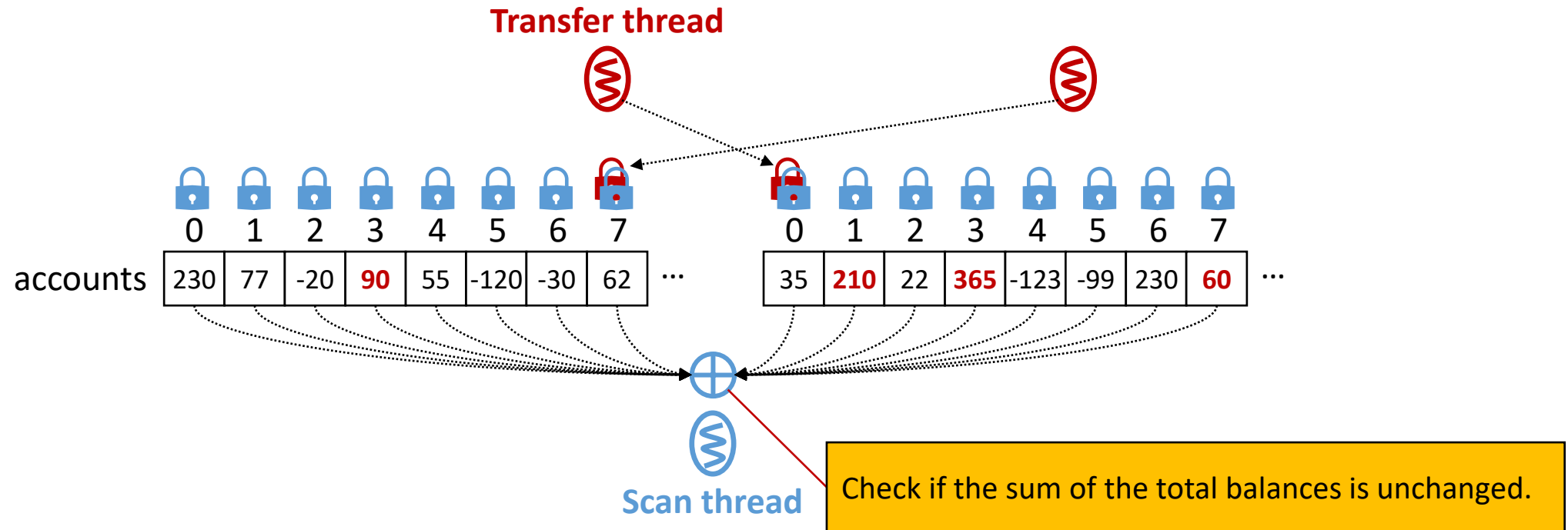
Test Code



Test Code



Test Code



Deadline & Regulations

- Deadline: **Nov 15 11:59pm**
- We'll only score your commit before the deadline, and your submission after the deadline will not be accepted.
- You must follow the given project hierarchy and the path of the “test_lock_table” executable file, otherwise, you cannot get a score.
- Without changing the contents of the main.cc file and the lock_table.h file, it should compile and run well.

Thank you