ECE650 Proj2 Report

1. Description of implementation

For both versions, I used: pthread_mutex_t lock = PTHREAD_MUTEX_INITIALIZER; to initialize a lock.

For lock version, I simply added the following two lines at the beginning and before the ending of the malloc and free function from the first homework.

```
pthread_mutex_lock(&lock);
pthread_mutex_unlock(&lock);

void *ts_malloc_lock(size_t size) {
pthread_mutex_lock(&lock);
...
pthread_mutex_unlock(&lock);
return ...;
}

void ts_free_lock(void *ptr) {
pthread_mutex_lock(&lock);
...
pthread_mutex_unlock(&lock);
}
```

The sections between pthread_mutex_lock and pthread_mutex_unlock are critical sections. This lock version provides a malloc and free level concurrency.

For no-lock version, I only added a lock before and after the sbrk function, and for the global variable head, I used local storage by using:

```
__thread ListNode* head;
```

The principle for this version is to create a free list for each thread. On every thread, codes are executed sequentially, so every free list is independent on each other, no overlapping memory region should appear. This no lock version provides a sbrk level concurrency.

```
  pthread_mutex_lock(&sbrklock);
  ... sbrk(...);
  pthread_mutex_unlock(&sbrklock);
  return ...;
```

2. Experiment Results:

Based on 10 tests for each test case, I got the following data:

Lock version pass rate

```
thread_test 10/10
thread_test_malloc_free 10/10
thread_test_malloc_free_change_thread 10/10
thread_test_measurement 10/10
```

No-lock version pass rate

```
thread_test 10/10
thread_test_malloc_free 10/10
thread_test_malloc_free_change_thread 10/10
thread_test_measurement 10/10
```

Baser on the 10 `thread_test_measurement` results, I made a form as the following:

	Lock version	No-lock version
Average Time	2.1s	0.36s
Maximum Time	3.1s	0.42s
Minimum Time	1.8s	0.33s
Average Size	42382034B	43265984B
Maximum Size	44271254B	44556324B
Minimum Size	41258455B	41234556B

Execution Time Analysis:

From the form, we can see that no-lock version is nearly five times the speed of lock version.

For lock version, all threads share one singly free list, the free list can be quite long, so it will take a long time to find the best fit one. For no-lock version, each thread has its own free list, so it will take much less time to find a fit node. Moreover, no-lock version would only lock during sbrk call, so for other operations like element adding or removal can happen simultaneously.

Data Segment Size Analysis:

Based on the results, there is not evident difference between two versions on size.

In conclusion, the no-lock version can improve the runtime while not make the data segment size larger.