1. Implementation Description

1.1 General Descriptions

For the assignment, we have to implement our version of malloc functions and free functions based on Best Fit Allocation, including:

```
//Thread Safe malloc/free: locking version
void *ts_malloc_lock(size_t size);
void ts_free_lock(void *ptr);

//Thread Safe malloc/free: non-locking version
void *ts_malloc_nolock(size_t size);
void ts_free_nolock(void *ptr);
```

1.2 Thread-Safe Malloc and Free with Locks

For the locking version of Thread-Safe malloc/free functions, we can utilize locks provided by the pthread library and needed synchronization primitives to support synchronization between different threads. The main idea is to lock the mutex right in the malloc function and unlock the mutex right before the malloc function returns, so we can make sure there's only one thread that can read and write the same code section at the same time. The critical section is between pthread_mutex_lock and pthread_mutex_unlock, and the other sections out of the critical section allow concurrency.

```
//Thread Safe malloc/free: locking version
void * ts_malloc_lock(size_t size) {
  // Acquire the lock
  pthread_mutex_lock(&lock);
  block * ans = bf_find(size);
  if (ans) {
    if (ans->size > size + HEADER_SIZE) {
      splitAndupdateBlock(ans, size);
    }
    else {
      removeBlock(ans);
    }
    void * data_ans = (void *)ans + HEADER_SIZE;
    // Release the lock
    pthread_mutex_unlock(&lock);
    return data_ans;
  }
  else {
    block * temp = extendBlock(size);
    if (temp == NULL) {
      pthread_mutex_unlock(&lock);
      return NULL;
    }
    void * data_cur = (void *)((unsigned long)temp + HEADER_SIZE);
    // Release the lock
    pthread_mutex_unlock(&lock);
    return data_cur;
  }
}
```

1.3 Thread-Safe Malloc and Free without Locks

For the no-locking version of Thread-Safe malloc/free functions, we can utilize Thread-Local Storage to support synchronization between different threads, and only use locks right before and after sbrk() function. The Thread-Local Storage can be used to allocate variables such that there is 1 per thread. By using the Thread-Local Storage, we can create and keep a Free List for each thread, so they will not affect others when reading and writing the data. The critical section is still between pthread_mutex_lock and pthread_mutex_unlock, and the other sections out of the critical section allow concurrency.

```
//Thread Safe malloc/free: non-locking version
void * ts_malloc_nolock(size_t size) {
  block * ans = bf_find_tls(size);
  if (ans) {
    if (ans->size > size + HEADER_SIZE) {
      splitAndupdateBlock_tls(ans, size);
    }
    else {
      removeBlock_tls(ans);
    }
    void * data_ans = (void *)ans + HEADER_SIZE;
    return data_ans;
  }
  else {
    block * temp = extendBlock(size);
    if (temp == NULL) {
      return NULL;
    }
    void * data_cur = (void *)((unsigned long)temp + HEADER_SIZE);
    return data_cur;
  }
}

void ts_free_nolock(void * ptr) {
  block * free_ptr = (block *)(ptr - HEADER_SIZE);
  addBlock_tls(free_ptr);
}
```

[Functions with *_tls are functions using variables created by TLS]

2 Result & Performance Analysis

2.1 Thread-Safe Malloc and Free with Locks

According to the results of testing of thread_test_measurement.c, the results of Thread-Safe Malloc and Free Functions with Locks are:

Average Execution Time/s	Average Data Segment Size/bytes
0.2146595	43001100

2.2 Thread-Safe Malloc and Free without Locks

The results of Thread-Safe Malloc and Free Functions without Locks are:

Average Execution Time/s	Average Data Segment Size/bytes
0.1746747	428655248

2.3 Performace Analysis

Based on the result of the tests, we can tell that the average execution time of thread-safe malloc and free implementation without locks is smaller than with locks, and the average data segment size is smaller as well. The lock method of one thread needs to require a lock before the critical section, and another thread that wants to access the critical section has to wait. So the execution time of the lock method is longer. When it comes to data segment size, the difference is small. If we can't find a free block to reuse in a free list through the lock method, it's a great possibility that we can't find one to reuse in multiple free lists through the no-lock method, and we still need to extend the size of the data segment.

2.4 Attachment

```
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.180270 seconds
Data Segment Size = 42824520 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.175221 seconds
Data Segment Size = 43481064 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.238856 seconds
Data Segment Size = 42905232 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.194030 seconds
Data Segment Size = 42737672 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.249632 seconds
Data Segment Size = 43099400 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.201322 seconds
Data Segment Size = 42252168 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.163771 seconds
Data Segment Size = 42135680 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.294313 seconds
Data Segment Size = 43964392 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.190297 seconds
Data Segment Size = 42538104 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.258883 seconds
Data Segment Size = 44072768 bytes
```

[Data of Thread-Safe Malloc and Free with Locks]

```
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.128784 seconds
Data Segment Size = 42802952 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
Segmentation fault (core dumped)
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.177144 seconds
Data Segment Size = 43087904 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.129200 seconds
Data Segment Size = 42733216 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.169137 seconds
Data Segment Size = 42152008 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.190558 seconds
Data Segment Size = 43025856 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.158355 seconds
Data Segment Size = 42473872 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.143744 seconds
Data Segment Size = 43522976 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.125286 seconds
Data Segment Size = 42745024 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.199471 seconds
Data Segment Size = 43080520 bytes
yy295@vcm-24549:~/ece650/650hw2/thread_tests$ ./thread_test_measurement
No overlapping allocated regions found!
Test passed
Execution Time = 0.131920 seconds
Data Segment Size = 43030920 bytes
```

[Data of Thread-Safe Malloc and Free without Locks]