1. Implementation

Locking version:

For the locking version, I just locked and unlocked bf_malloc() and bf_free() functions. In this way, only one thread could malloc and free memory, other threads must wait until it's done. The rest of my code is the same as hw1.

Non-locking version:

For the non-locking version, I used Thread-Local Storage variables to keep track of the head and tail of freelist. If I create a variable in TLS, every thread has its own copy of the variable. In this way, even if multiple threads enter the critical section at the same time, since the variables are in TLS, they would all have their own copy of the memory, each thread would have their own freelist, so they would not race with each other.

For both of the versions, critical sections refers to memory malloc and free part of my code.

2. Results from experiment

Experiments	Locking Version		Non-Locking Version	
	Execution time/s	Data Segment Size/bytes	Execution Time/s	Data Segment Size/bytes
1	0.123569	45140608	0.092926	43876648
2	0.187655	45852376	0.176566	44029456
3	0.181564	45447576	0.163697	44332248
4	0.126892	45322704	0.114014	44436784
5	0.130583	44273096	0.124246	44212296

Locking version average execution time = 0.150052s

Locking version average data segment size = 45207272 bytes

Non-locking version average execution time = 0.134289s

Non-locking version average data segment size = 44177486 bytes

In the locking version, all threads must enter the critical section one by one, so its execution time is longer. The data segmentation size of non-locking version is a bit smaller, but almost the same to the locking version.