Firstly, I created a node struct to keep id,size and index values of a node of the linked list. Then I determined the private members that will be used. These are:

1) list<node> heap_list → the linked list which simulates the heap.
2) İnt init_size → stores the total size of the heap
3) pthread_mutex_t mutex_for_allocations → the mutex which is used for myMalloc and myFree functions
4) pthread_mutex_t mutex_for_print → the mutex which is used for myMalloc and myFree functions

Basically,mutexes are locked at the beginning of the functions and are unlocked at the end of the functions. Therefore, each thread will use functions one by one and thus, synchronization is achieved.

In initHeap function, a single node that represents the linked list is created. Mutexes are also initialized.

In myMalloc function, an iterator starts from the beginning of the linked list to traverse the linked list to find a node which is free and whose size is larger than the node that will be allocated. If an appropriate node is found, the allocation operation will be done. Otherwise, a message which indicates that allocation cannot be done is printed.

In MyFree function, I checked whether size of the linked list is 1. If the size is 1 ,then it means that we have not any allocated node, so no free operation can be done. Then, I checked three conditions:

1)node that will be freed is the first node

2) node that will be freed is the last node

3) node that will be freed is somewhere in the middle

After that, delete operations are done. Also, if coalescing is possible for any of these three cases, it is done too.