Programming Assignment (PA) - 4 Heap Management
Yigit Kaan Tonkaz
29154

First of all, I implement a struct called node to use in my LinkedList implementation. It has id, size, index and next pointer properties. And I created Heap Manager class with its methods and constructor. In constructor, my head pointer and pthread are initialized.
Then I implement the method functions,

initHeap
I created a new node to start and make the head of my LinkedList. And I print it.

myMalloc
Firstly, I lock my pthread to avoid conflicts between threads. I create 2 pointers each time when myMalloc called which are current and prev pointers to be able to travel and make changes in my LinkedList. I used while loop to check each node whether they are suitable in terms of bytes and if greater or equal to current's node size, it goes in my if or else if statements, and create a new node for the given size and add it to my LinkedList with connecting the next pointers. Also I keep my return value in variable called ret which is initialized -1. If the inner if statements are correct ret value changed but If they not , while loops travel the LinkedList and eventually finished and print "Cannot allocate". At the end I unlock pthread to continue the next thread.

myFree
myFree also has similar implementation structure with myMalloc, but it checks whether the current node has right or left node with id -1. If yes it deletes the current node and add the size to other(prev) node. In other words, it connects the nodes and delete the current one (adding values to previous one in both right and left cases). Also it checks the return value and if inner if statements are correct(either of them), they change the ret value to 1 which blocks the "Cannot deallocate" statement.

Print
I create temporary pointer to print each node with the correct format.

In short, I implement my functions and LinkedList with struct structure and used pointers to travel between nodes or make changes on them. For the concurrency, I used pthread and lock, unlock system in the myMalloc and myFree method functions because threads may deadlock or overlap or interrupting each other while try to start these functions, also at the end of the functions. Therefore, I lock them at the beginning and unlock them at the end which solves the concurrency problem.