

BITS Pilani, Pilani Campus  
2<sup>nd</sup> Sem. 2020-21  
CS F211 Data Structures & Algorithms

=====

Lab 8

=====

**Instructions**

- In this lab you have to solve two problems.

**Problem 1: [Expected Time: 75 minutes.]**

An array of student records *studArray*, where each record is a tuple – {<name>, <marks>} is defined in the file named *main.c*. <name> is a string of size 10 and <marks> is a *float* value. You can use the files in folder **q1** and build your code over them. **DO NOT** make any changes to the **structure definitions** and **signatures** and **return types** of the predefined functions in these files. However, you may add more functions as required.

- a. Implement the function *createList()*, which takes an array of student records along with its size, creates a linked list of records contained in the array and returns the list. This function **must** execute in  $O(n)$  time, where  $n$  is the size of the array.
- b. Implement the function *insertInOrder()*, which takes a sorted linked list and a list node containing a new student record and inserts it into the list at its appropriate place such that the list remains sorted in *ascending order* by <marks>.
- c. Implement the function *insertionSort()*, which takes an unsorted list and sorts it using the *insertInOrder()* function and returns the sorted list.
- d. Implement a function *measureSortingTime()*, which takes an unsorted list and sorts it by calling *insertionSort()* and returns the time taken for the execution of *insertionSort()* in *milliseconds*.
- e. Create your custom *myalloc()* and *myfree()* functions to replace calls to native *malloc()* and *free()* functions to profile the heap memory. The total memory allocated at any given time should be stored in a global variable known as *"globalCnt"*.

**Problem 2: [Expected Time: 75 minutes]**

Your input file contains the number of records in its first line and one student record per subsequent line. Each record has two fields separated by one space character: <name> a string of max size 10 and <marks> a *int* value. You can use the files in folder **q2** and build your code over them. A sample input file – *Input.txt* is also provided in the zip. **DO NOT** make any changes to the **signatures** and **return types** of the predefined functions in these files. You may also add more functions as required.

- a. Implement a function – *"readData()"*, that takes in a string containing the file name, reads the file into an array of integer values containing the marks only. This function should also store the size of the array in a global variable called *size* (declared in *"qsort.h"*). Don't change the signature and return type of this function.
- b. Implement a  $\theta(n)$ -time - *"extractKeys()"* function that takes an array *Ls* of  $n$  integer records, the size  $n$ , and finds all the keys in it and returns them in a sorted array named *Keys* along with its size. Keys should not contain any duplicates. It is known

that each record in  $L_s$  will have a key in the range  $loKey..hiKey$  (taken as inputs to the function) and all values in this range need not occur as a key.

- c. Implement a  $\theta(n)$ -time locality-aware *partitioning* function – “part2Loc()” that takes an array  $L_s$  of  $n$  integers, the lower index  $lo$  and higher index  $hi$  of  $L_s$ , and an integer pivot (key) as arguments and partitions  $L_s$  into two sub-lists based on the pivot.