Lab #9

CS-2050

November 4, 2022

1 Requirements

In this lab, you will cover sorting and searching arrays. In this lab, every function has an associated complexity in the form of Big-O notation. Each function must perform at the complexity specified, and must not perform faster or slower than required. The method by which you fulfill the lab requirements is not defined, and you must derive the implementation from the complexity requirements listed above each function. For any details of the implementation not specified in the requirements, you are free to make your own design decisions. It is a good idea to use helper functions to complete this lab. You are given the following struct definition in your starter code:

```
typedef struct {
        int age;
        float height;
        double kilometers;
} Runner;
```

1.1 sortByKM

```
// O(n^2)
void sortByKM(Runner *array, int size);
```

Info: This function will take an array of Runner structs, as well as the size of the array. It will sort the array in ascending order based upon the **kilometers** member of each element. You may assume the array will contain duplicate data, how you order duplicate elements is up to you.

1.2 getByKM

```
// O(log(n))
int getByKM(Runner *array, int size, double kilometers);
```

Info: This function will take an array of Runner structs sorted in ascending order by kilometers, and the size of the array. It will efficiently search the array for an element with the kilometers given, and return the index of that element, or -1 if one was not found.

Notice

Grading: 20 points

1. Write required *sort* function
   * 10 points
2. Write required *get* function
   * 10 points