Lab #9

CS-2050

April 6, 2023

1 Requirements

In this lab, you will cover executing an efficient search on a sorted array. Remember that not all of the lab's requirements may be possible without the use of helper functions. In this lab you are given the following struct definition:

```
typedef struct {
        unsigned long accountNumber;
        float accountWorth;
        int daysActive;
} Client;
```

1.1 makeArray

```
void * makeArray(int arraySize, int elementSize)
```

Info: This function will take an array size, as well as the size of each element in the array. It will allocate an array with the given size, and store the size before the start of the array as an int. If allocating the array was successful, it will return a pointer to the array, otherwise it will return NULL.

1.2 getSize

```
int getSize(void *array)
```

Info: This function takes an array which was allocated with makeArray, and returns the size stored before the array.

1.3 searchClients

```
// O(log(n))

int searchClients(Client *array, Client *query)
```

Info: This function performs a *binary search* on the given struct array using **recursion**. This function will return the index of the query struct when it is located, or **-1** on error.

1.4 compareClients

```
// O(1)

int compareClients(Client *a, Client *b)
```

Info: This function compares the two structs given by their **accountWorth** members. It should return a *strictly negative* number if $a < b$, a *strictly positive* number if $a > b$, or 0 if they are equal.

1.5 freeArray

```
void freeArray(void *array)
```

Info: This function takes an array which was allocated with makeArray, and frees the memory allocated to the array.

Grading: 15 points

1. Write required *makeArray* function
   - 1 points
2. Write required *getSize* function
   - 1 points
3. Write required *searchClients* function
   - 10 points
4. Write required *compareClients* function
   - 2 points
5. Write required *freeArray* function
   - 1 points

Notice:

1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the $-Wall$ and $-Werror$ flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab policy document.