# Lab #9
Summer 2024

## 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:

```
struct _Computer
{
    char     sCompany[MAXSTRING];     // Manufacturing company (e.g., Dell)
    char     sModel[MAXSTRING];       // (e.g., "XPS13")
    float    fltClockSpeed;           // Clock speed in HZ
    int      iDisk;                   // Disk size in bytes
    int      iMemory;                 // Memory size in bytes
};
typedef struct _Computer Computer;
```

### 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 searchComputersByDiskSize

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

int searchComputersByDiskSize(Computer *array, Computer *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 (0-based) when it is located, or **-1** on error.

### 1.4 compareComputersByDiskSize

```
// O(1)
int compareComputersByDiskSize(Computer *a, Computer *b)
```

**Info:** This function compares the two structs given by their **iDisk** 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.

## Description

To get started on this lab, type the following while logged in to hellbender.rnet.missouri.edu:

```
cs2050start lab9
cd lab9
make
./a.out
```

For the lab assignment, you are to start with the starter code provided in the lab9 directory. This starter code is an outline of what you need to do. This time, you will need to change lab9.c as well as lab9main.c (since no tests are provided). You only need to submit lab9.c.

## Bonus Points Opportunity

You may write functions searchComputersByCompanyAndModel() and compareComputersByCompanyAndModel() for bonus points. These are similar to the required functions, but search or compare respectively the structures by the sCompany and sModel members. Note that you would only need to look at the sModel member if the sCompany member were identical in the structures you are considering. **Hint:** Take a look at the strcmp() Standard C Library function if you don't remember it from your CMP_SC 1050 days.

In order to get credit, searchComputersByCompanyAndModel() must be implemented using recursion and take O(log(n)) complexity. compareComputersByCompanyAndModel() must take O(1) complexity.

## Submission Information

Submit this assignment by using the mucsmake command on Hellbender to submit lab9.c:

```
mucsmake 2050 lab9 lab9.c
```

## Rubric

1. Write required *makeArray* function
   * 1 point
2. Write required *getSize* function
   * 1 point
3. Write required *searchComputersByDiskSize* function
   * 10 points
4. Write required *compareComputersByDiskSize* function
   * 2 points
5. Write required *freeArray* function
   * 1 point
6. Write optional *searchComputersByCompanyAndModel* function
   * 4 points
7. Write optional *compareComputersByCompanyAndModel* function
   * 1 point

## Notice:

1. All of your lab submissions must include documentation in the form of code comments to receive full points. In addition, your program is expected to have **a comment header** at the top that includes your name, pawprint, the course you are taking, and the lab that you solved.
2. All of your lab submissions must compile under GCC using the −*Wall* and −*Werror* flags to be considered for a grade (note that the Makefile should take care of this for you in this case).
3. Do **NOT** change any files other than lab9.c and lab9main.c. Additional helper functions must be placed in your lab9.c file.