Lab #2

CS-2050

February 2, 2024

1 Requirements

In this lab, you will cover using double pointers, and dynamically allocating memory. Unless otherwise stated, you should assume that any arrays received as parameters are **not sorted**. Note that you may find some of your required functions helpful in implementing the others.

1.1 makeArray

```
int * makeArray(int size)
```

Info: This function will take an integer representing the size of a new int array to create. It will create the new array, and initialize each index of the array to **0**. On success, it returns a pointer to the newly allocated and initialized array, or NULL on failure.

1.2 addressOf

```
int * addressOf(int *array, int size, int element)
```

Info: This function will take an int array, as well as the size of the array, and a query element. It will find and return the address of the query element in the array, or NULL if it does not exist. You may assume there are no duplicates in the array.

1.3 sliceArray

```
int sliceArray(int *array, int size, int begin, int end, int **result)
```

Info: This function takes an int array, the size of the array, and two boundary elements. It will subsection the array around the two boundary elements, such that the start of the new array includes the begin element, and the end of the new array includes the end element (at newSize - 1). This is referred to as a "slice" operation. The result pointer should be updated to point to the start of the new array, and this function should return the size of the new array.

You may assume that so long as both boundary elements exist in the array, that the begin will appear **before** the end. If **either** boundary element is not found in the array, then this function returns -1 and does not perform the slice operation. Note that you **DO NOT** need to allocate memory for this operation, as both arrays share the same block of memory. Example:

```
array = { 2, 9, 4, 3, 0, 7, 1 }
// After calling sliceArray with begin=9, end=0
slice = { 9, 4, 3, 0 }
```

1.4 freeArray

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

Info: This function takes an array and frees it. It should set the referenced pointer to NULL after freeing.

Grading: 11 points

1. Write required *makeArray* function
   * 2 points
2. Write required *addressOf* function
   * 2 points
3. Write required *sliceArray* function
   * 5 points
4. Write required *freeArray* function
   * 2 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 $-Wall$, $-Werror$, and $-Wpedantic$ 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 intro PDF.