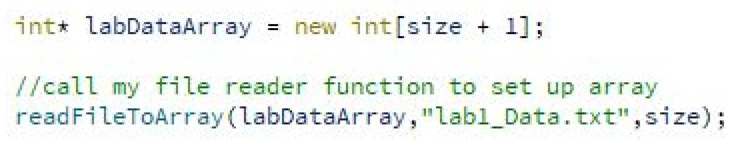
Reggie Johnson
CS 361- Algorithms

1. When reading in the data file I made a header file that held a file reader function that is used to read integers into an array. It takes in an int that determines the number of files to read in. When timing the function the following table was produced. The time complexity for this algorithm



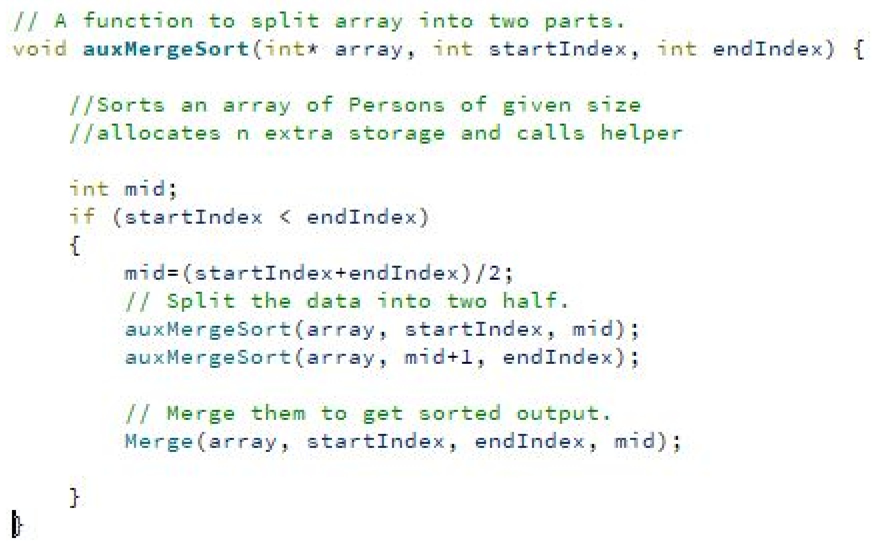
```
int* labDataArray = new int[size + 1];

//call my file reader function to set up array
readFileToArray(labDataArray,"lab1_Data.txt",size);
```

Reading in file --- O(n) for the while loop to read each element of the array.
Create 2 ints, an array, a long long int -- O(1) x 4
Add sum of current value to sum --- O(1)
We cout to the console, close our input file -- O(1) x 3

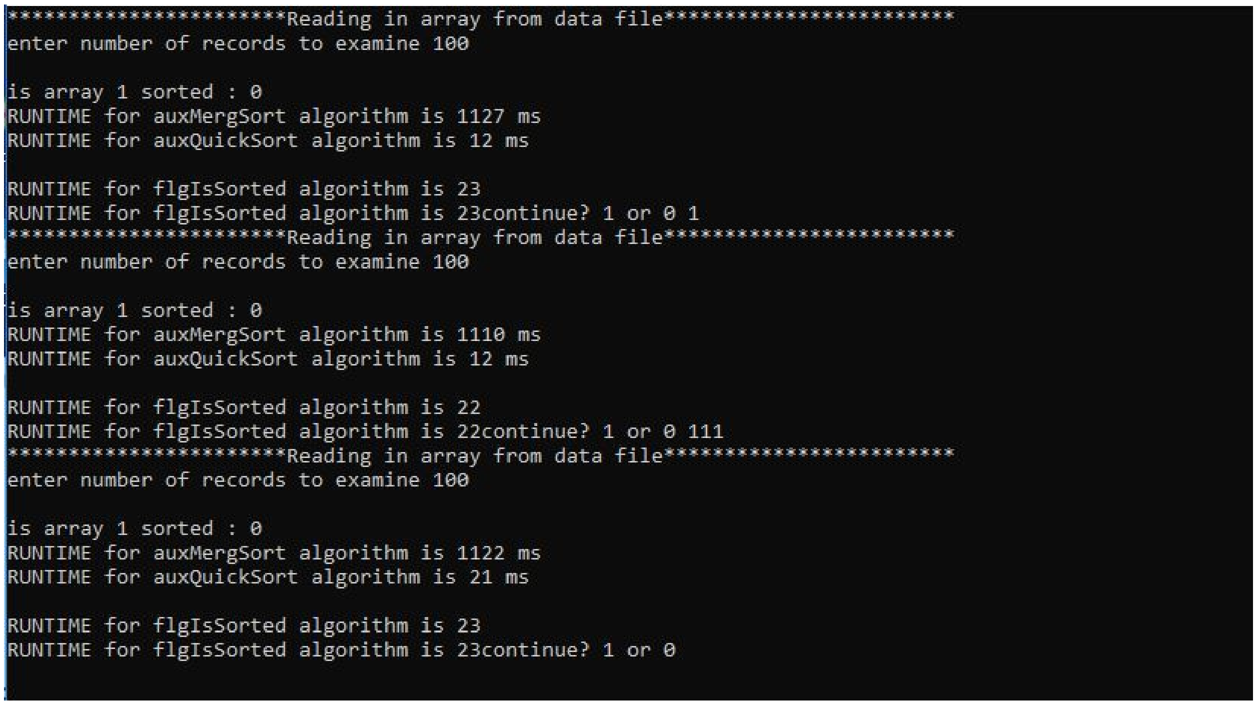
Should be O(n)

For timing I used the chronos header file to measure in milliseconds. For the smaller sized arrays, I looped the functions by a factor of 100 to estimate the execution time I used the calculation RUNTIME/100. Arrays sizes 10000 and lower looped 1000 times.

2. When implementing merge sort I used a helper function called Merge which takes the two subarrays and merges them into one. The divide step computes the midpoint of each of the sub-arrays. Each of this step just take O(1) time. The conquer step recursively sorts two subarrays of n/2 (for even n) elements each. I used my merge sort function from last year and re coded it to work with int arrays.



```
// A function to split array into two parts.
void auxMergeSort(int* array, int startIndex, int endIndex) {

    //Sorts an array of Persons of given size
    //allocates n extra storage and calls helper

    int mid;
    if (startIndex < endIndex)
    {
        mid=(startIndex+endIndex)/2;
        // Split the data into two half.
        auxMergeSort(array, startIndex, mid);
        auxMergeSort(array, mid+1, endIndex);

        // Merge them to get sorted output.
        Merge(array, startIndex, endIndex, mid);

    }
}
```



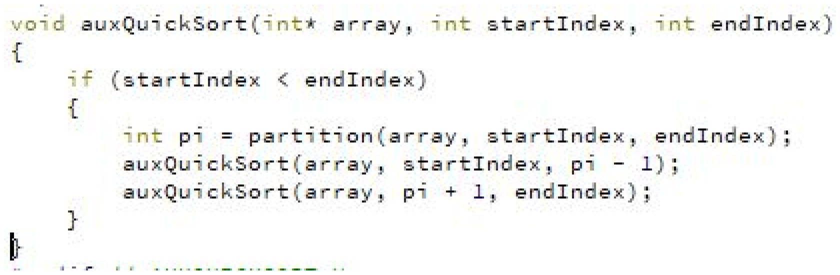
```
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1127 ms
RUNTIME for auxQuickSort algorithm is 12 ms

RUNTIME for flgIsSorted algorithm is 23
RUNTIME for flgIsSorted algorithm is 23continue? 1 or 0 1
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1110 ms
RUNTIME for auxQuickSort algorithm is 12 ms

RUNTIME for flgIsSorted algorithm is 22
RUNTIME for flgIsSorted algorithm is 22continue? 1 or 0 111
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1122 ms
RUNTIME for auxQuickSort algorithm is 21 ms

RUNTIME for flgIsSorted algorithm is 23
RUNTIME for flgIsSorted algorithm is 23continue? 1 or 0
```

T(n) = 2T(n/2) + ɵ(n) --->
n^ log2 2 --->
n^1 = n====>

according to the master theorem since n^logbA = f(n) then the time complexity is

n^log2 2 * logn which equals big O(n*logn)

3. For quick sort I used one helper function called partition that sorts the array into 2 subarrays and runs quicksort on each of them.

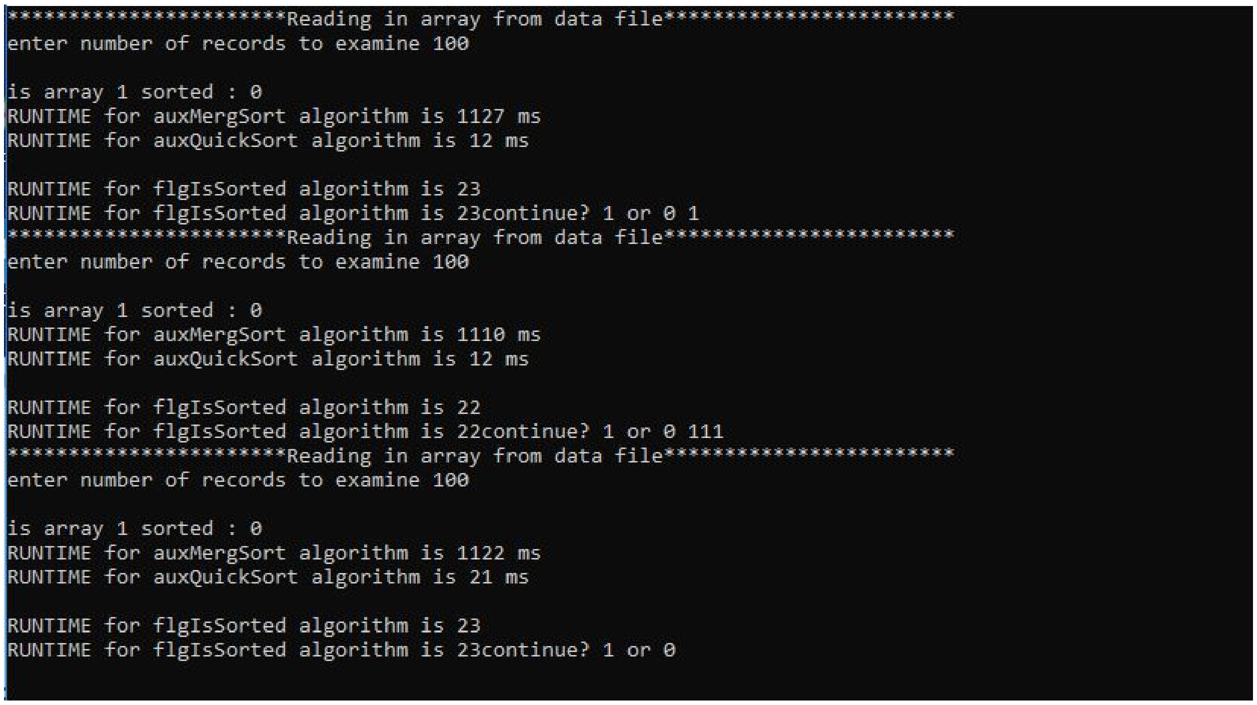


```
void auxQuickSort(int* array, int startIndex, int endIndex)
{
    if (startIndex < endIndex)
    {
        int pi = partition(array, startIndex, endIndex);
        auxQuickSort(array, startIndex, pi - 1);
        auxQuickSort(array, pi + 1, endIndex);
    }
}
```

Function,
2T(n/2) + theta(n)
a = 2, b = 2

Reggie Johnson
CS 361- Algorithms

Here,



```
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1127 ms
RUNTIME for auxQuickSort algorithm is 12 ms

RUNTIME for flgIsSorted algorithm is 23
RUNTIME for flgIsSorted algorithm is 23continue? 1 or 0 1
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1110 ms
RUNTIME for auxQuickSort algorithm is 12 ms

RUNTIME for flgIsSorted algorithm is 22
RUNTIME for flgIsSorted algorithm is 22continue? 1 or 0 111
***********************Reading in array from data file************************
enter number of records to examine 100

is array 1 sorted : 0
RUNTIME for auxMergSort algorithm is 1122 ms
RUNTIME for auxQuickSort algorithm is 21 ms

RUNTIME for flgIsSorted algorithm is 23
RUNTIME for flgIsSorted algorithm is 23continue? 1 or 0
```

Theta (n^log2 2) = theta(n)(2nd condition)
So the complexity will be: bigO(nlogn)complexity

To create the quick sort function I googled quicksort nad found geeksforgeeks example. I also watch a few youtube videos, one from HackerRank, to learn to code the algorithm. Quick sort performed faster than mergesort on values lower than 100000

4. flgIsSorted was the most difficult for to implement. I created 2 helper functions to pass the array into. Due to the large size of the array it was necessary to check if the array is sorted piece by piece. I picked an arbitrary number, 10,000, to do it.

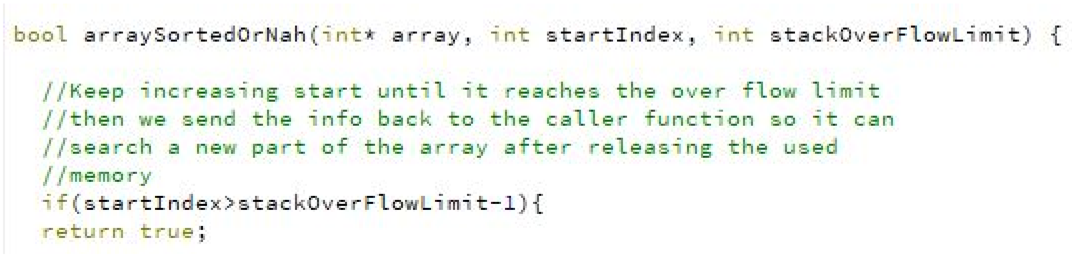
flgIsSorted passes the array and its size to the helper function isSortedArray.which is O(1);

isSorted goes through the entire array and compares one index to another for comparison. IF size is bigger htan the arbitruary threshold, then a 2nd helper function is called, simply to check the array 10,000 elements at a time. When coming up with the code I searched online and found an article on geeksforgeeks outlining code for check array.

https://www.geeksforgeeks.org/program-check-array-sorted-not-iterative-recursive/

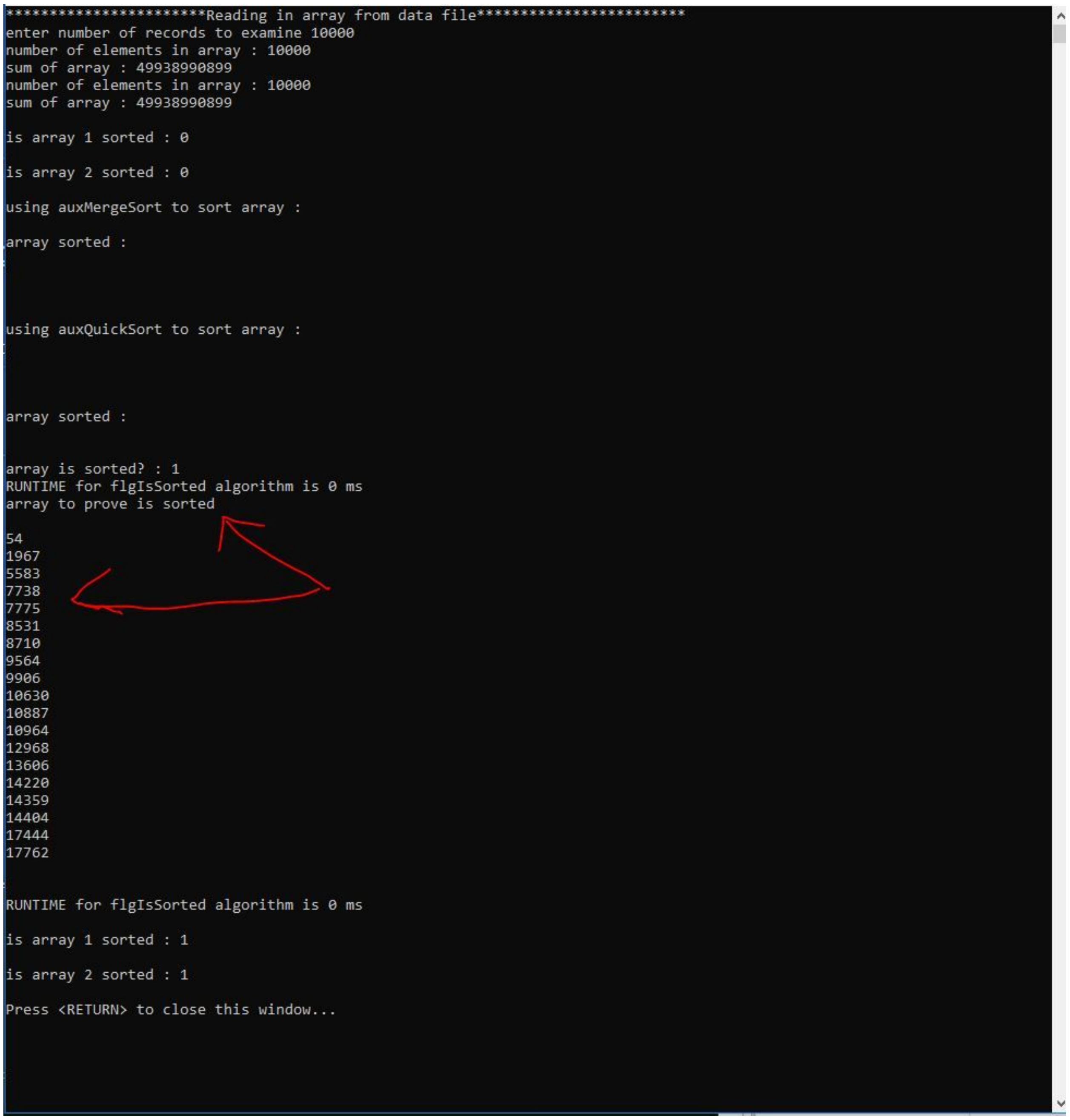
I used a little bit of the code but I had to modiy it to work around the stackOverFlow

Reggie Johnson
CS 361- Algorithms



```
bool arraySortedOrNah(int* array, int startIndex, int stackOverFlowLimit) {

  //Keep increasing start until it reaches the over flow limit
  //then we send the info back to the caller function so it can
  //search a new part of the array after releasing the used
  //memory
  if(startIndex>stackOverFlowLimit-1){
  return true;
```

flgIsSorted tested

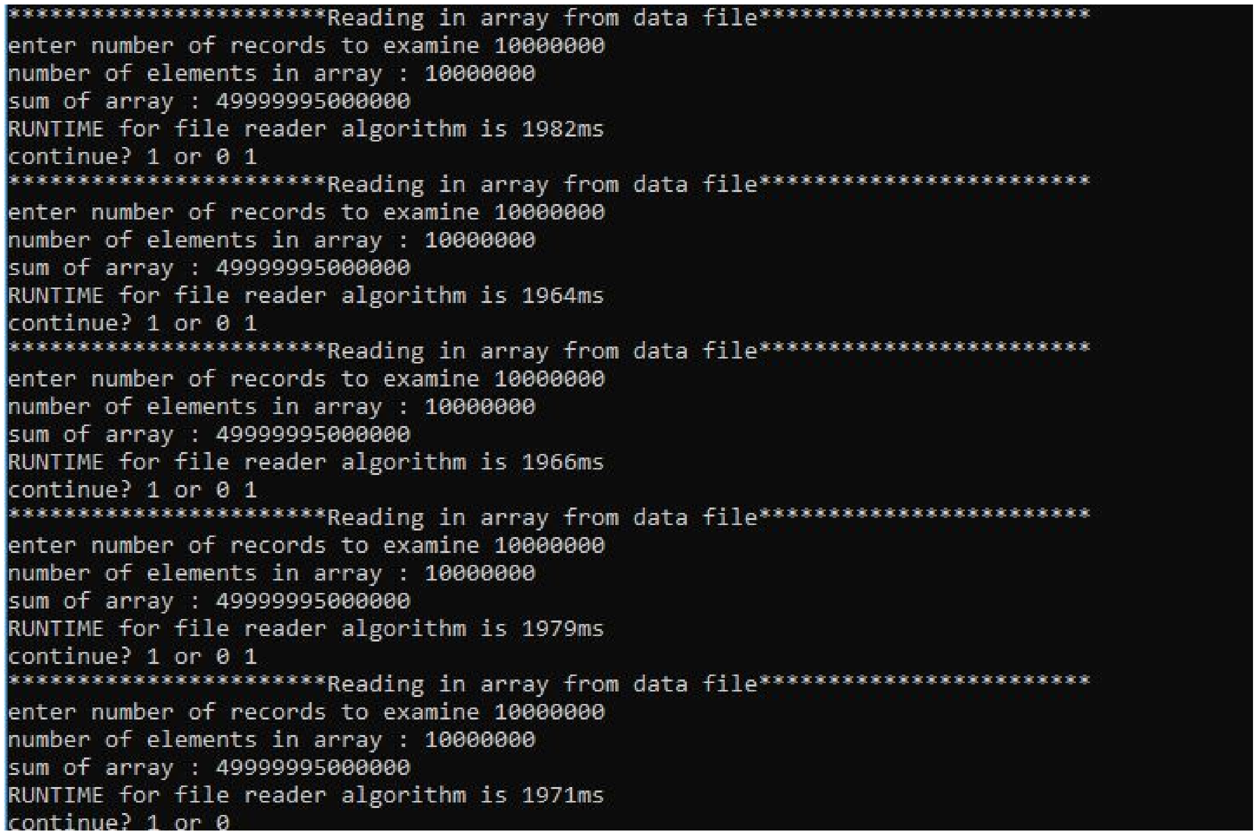


```
***********************Reading in array from data file************************
enter number of records to examine 10000
number of elements in array : 10000
sum of array : 49938990899
number of elements in array : 10000
sum of array : 49938990899

is array 1 sorted : 0

is array 2 sorted : 0

using auxMergeSort to sort array :

array sorted :



using auxQuickSort to sort array :




array sorted :


array is sorted? : 1
RUNTIME for flgIsSorted algorithm is 0 ms
array to prove is sorted

54
1967
5583
7738
7775
8531
8710
9564
9906
10630
10887
10964
12968
13606
14220
14359
14404
17444
17762


RUNTIME for flgIsSorted algorithm is 0 ms

is array 1 sorted : 1

is array 2 sorted : 1

Press <RETURN> to close this window...
```

Reggie Johnson
CS 361- Algorithms

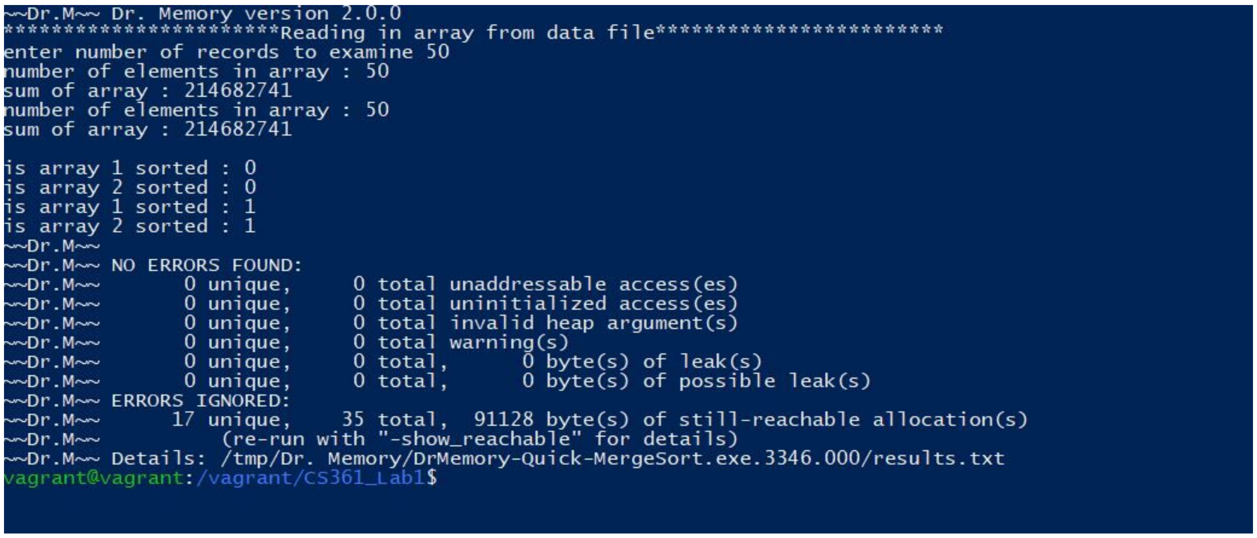
Screen dumps

Reading in file run time



```
************************Reading in array from data file*************************
enter number of records to examine 10000000
number of elements in array : 10000000
sum of array : 49999995000000
RUNTIME for file reader algorithm is 1982ms
continue? 1 or 0 1
************************Reading in array from data file*************************
enter number of records to examine 10000000
number of elements in array : 10000000
sum of array : 49999995000000
RUNTIME for file reader algorithm is 1964ms
continue? 1 or 0 1
************************Reading in array from data file*************************
enter number of records to examine 10000000
number of elements in array : 10000000
sum of array : 49999995000000
RUNTIME for file reader algorithm is 1966ms
continue? 1 or 0 1
************************Reading in array from data file*************************
enter number of records to examine 10000000
number of elements in array : 10000000
sum of array : 49999995000000
RUNTIME for file reader algorithm is 1979ms
continue? 1 or 0 1
************************Reading in array from data file*************************
enter number of records to examine 10000000
number of elements in array : 10000000
sum of array : 49999995000000
RUNTIME for file reader algorithm is 1971ms
continue? 1 or 0
```

Drmemory to check for leaks, array accuracy and isSorted Function working



```
~~Dr.M~~ Dr. Memory version 2.0.0
************************Reading in array from data file*************************
enter number of records to examine 50
number of elements in array : 50
sum of array : 214682741
number of elements in array : 50
sum of array : 214682741

is array 1 sorted : 0
is array 2 sorted : 0
is array 1 sorted : 1
is array 2 sorted : 1
~~Dr.M~~
~~Dr.M~~ NO ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       0 unique,     0 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
~~Dr.M~~ ERRORS IGNORED:
~~Dr.M~~      17 unique,    35 total,  91128 byte(s) of still-reachable allocation(s)
~~Dr.M~~          (re-run with "-show_reachable" for details)
~~Dr.M~~ Details: /tmp/Dr. Memory/DrMemory-Quick-MergeSort.exe.3346.000/results.txt
vagrant@vagrant:/vagrant/CS361_Lab1$
```