CSCI 200 - Fall 2023 Foundational Programming Concepts & Design

Lab 6B - The Needle in the Haystack

This lab is due by Thursday, December 07, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

Jump To: **Rubric Submission**

Sorting Instructions

For this lab, add to your abstract `List` class a purely virtual `void sort()` method. This function will need to be implemented as both `Array::sort()` and `LinkedList::sort()` . Both of these implementations will perform the recursive Merge Sort algorithm.

To test your implementations, perform the following steps for each `List` implementation:

1. Create a list of integers
2. Populate the list with the values in order: 4 3 8 1 5 9 7 2 6
3. Print the list forwards (prints 4 3 8 1 5 9 7 2 6)
4. Sort the list
5. Print the list forwards (prints 1 2 3 4 5 6 7 8 9)

To assist with the debugging of the recursive steps, have each function call print out the contents of the left and right lists after splitting. Additionally, print out the contents of the resultant merged list each function call.

You are encouraged to test your sorting with other list contents to verify its correctness (such as a list of strings).

An example run of the program may be:

```
Sorting an array:
Initial array: 4 3 8 1 5 9 7 2 6
Left:  4 3 8 1 5
Right: 9 7 2 6

Initial array:  4 3 8 1 5
Left:  4 3 8
Right: 1 5

Initial array: 4 3 8
Left:  4 3
Right: 8

Initial array: 4 3
Left:  4
Right: 3

Merged: 3 4

Merged: 3 4 8

Initial array: 1 5
Left:  1
Right: 5

Merged: 1 5

Merged: 1 3 4 5 8

Initial array: 9 7 2 6
Left:  9 7
Right: 2 6

Initial array: 9 7
Left:  9
Right: 7

Merged: 7 9

Initial array: 2 6
Left:  2
Right: 6

Merged: 2 6

Merged: 2 6 7 9
```

```
Merged: 1 2 3 4 5 6 7 8 9
```

When it's verified the sorting algorithm is operating correctly, remove the debugging outputs to clean up the generated output. Now the example run would be similar to below:

```
Sorting an array:
Initial array: 4 3 8 1 5 9 7 2 6
Sorted array: 1 2 3 4 5 6 7 8 9

Sorting a Linked List:
Initial list: 4 3 8 1 5 9 7 2 6
Sorted list: 1 2 3 4 5 6 7 8 9

Sorting a Linked List:
Initial list: dog cat bird elephant
Sorted list: bird cat dog elephant
```

Searching Instructions

Next, add to your abstract `List` class a purely virtual `int search(T)` method. This function will need to be implemented as both `Array::search(T)` and `LinkedList::search(T)`. The Array implementation will use binary search while the LinkedList implementation will use linear search.

To test your implementations, perform the following steps:

1. Ask the user which list implementation to use
2. Ask the user how many integers to enter `n`
3. Ask the user for the smallest value to generate `min`
4. Ask the user for the largest value to generate `max`
5. Create a list of `n` integers
6. Assign each element a random value within the range provided `[min, max]`
7. Print the list forwards
8. Sort the list ascending
9. Print the list forwards
10. Ask the user how many target values they wish to search for
11. For each target value entered by the user, perform an iterative binary search to find the target. If the target is found, then print the position the target is first found at. If the target is not found, then print -1.

For example, if our sorted array contains `1 3 3 3 5` and the user is searching for

- `1` , then the value is found at position 0
- `3` , then the value is found at position 2
- `5` , then the value is found at position 4
- `2` , then the value is "found" at position -1

For example, if our sorted linked list contains `1 3 3 3 5` and the user is searching for

- `1` , then the value is found at position 0
- `3` , then the value is found at position 1
- `5` , then the value is found at position 4
- `2` , then the value is "found" at position -1

Grading Rubric

Your submission will be graded according to the following rubric:

Points	Requirement Description
0.70	Fully meets specifications
0.15	Submitted correctly by Thursday, December 07, 2023, 11:59 PM
0.15	**Best Practices** and **Style Guide** followed
1.00	**Total Points**

Lab Submission

Always, **always**, **ALWAYS** update the header comments at the top of your `main.cpp` file. And if you ever get stuck, remember that there is LOTS of **help** available.

Zip together your `main.cpp, Makefile, List.hpp, Array.hpp, LinkedList.hpp` files and name the zip file `L6B.zip` . Upload this zip file to Canvas under L6B.

This lab is due by Thursday, December 07, 2023, 11:59 PM.
As with all labs you may, and are encouraged, to pair program a solution

to this lab. If you choose to pair program a solution, be sure that you individually understand how to generate the correct solution.

[Jump to Top] [Site Map]