Lab 03 - Divide & Conquer

Instructions:

- **When a dataset is sorted, it can be searched more efficiently using the binary search algorithm. Binary search begins by comparing the target value to the middle element of the dataset.**
  - **If the target matches the middle element, the algorithm returns its index.**
  - **If the target is less than the middle element, the search continues in the lower half of the dataset.**
  - **If the target is greater than the middle element, the search continues in the upper half.**

  This process repeats until the target is found or the dataset can no longer be divided. If the value is not present, the algorithm typically returns -1 (or another indicator such as the dataset's size) to signal that the target was not found.

- **Your objective is to construct a binary search algorithm, determine its runtime, and test it.**

- **Your source codes must compile and can only include the libraries '`iostream`', '`iomanip`', '`string`', and user-defined libraries from the lab to receive any credit.**

- **A cumulative task will not receive credit if the required previous tasks are not completed.**

- **Your submissions must be submitted to the GitHub repository in the Lab03 directory.**

- **Cheating of any kind is prohibited and will not be tolerated.**

- **Violating or failing to follow any of the rules above will result in an automatic zero (0) for the lab.**

Grading

Task	Maximum Points	Points Earned
1	1	
2	1	
3	1	
4	1	
5	1	
Total	5	

Note: solutions will be provided for tasks colored blue only.

Task 1

- Create a text file named ‘`pseudocode.txt`’ that includes the pseudocode for your recursive binary search algorithm using the pseudocode syntax instructions provided in class. The algorithm should take an array, two endpoints, and a target as inputs.

Task 2

- Create a text file named ‘`runtime.txt`’ that states the recurrence function of your algorithm from Question 1, and determines and proves its runtime using the master theorem.

Task 3

- Create a header file named ‘`Algorithm.h`’ that defines a C++ function of your pseudocode from Question 1 and a sorting algorithm function.

Task 4

- Create a text file named ‘`simulate.txt`’ to simulate the execution of your algorithm from Question 1 on the following dataset:

  `[15, 19, 24, 29, 32, 33, 39, 41, 45, 49, 53, 58, 60, 66, 70, 74]`

  Simulate each of the target values 15, 74, 64, and 20. For each target:

  - Show every midpoint value chosen during each recursive call by enclosing it in parentheses.
  - Conclude each simulation with the final output.

  Example:

  For dataset `[12, 34, 48, 56, 67, 72, 81]` and target 67, the simulation would be

  1. `[12, 34, 48, (56), 67, 72, 81]`
  2. `[12, 34, 48, 56, 67, (72), 81]`
  3. `[12, 34, 48, 56, (67), 72, 81]`
  4. `output: 4`

Task 5

- Create a C++ file named ‘`test.cpp`’ that includes ‘`Algorithm.h`’ and verifies the outputs obtained in Question 4. Next, demonstrate a binary search on an unsorted array using a target value that exists in the dataset. Afterward, sort the array and perform the same binary search call again using the same arguments, then display both results for comparison.

Extra Credit

- Create a text file named ‘`extra.txt`’ that proves

$$\frac{4}{3}n^2 - 4n + 6 = O(n^2)$$

  by finding constants c and n_0 that satisfy the big-oh notation definition.

  (0.5 point)