Problem One: File Processing and Sorting

Write a function that takes as input the name of a file containing a list of integers, one per line, then returns the median value in the file. If the file is empty or does not exist, your function should have appropriate behavior.

Problem Two: Linked Lists

Write a function in a programming language of your choice that accepts a linked list as input, then reverses that linked list. If the programming language you choose has a library linked list type (e.g. `std::list` or `ArrayList`), we'd prefer that you not use it and instead create your own type representing a linked list. What is the time complexity of your solution (O(n)? O(n log n)? O(n^2)? Something else?)

Problem Three: String Algorithms

In a programming language of your choice, write a function that accepts as input two strings, a text string and a pattern string, then returns the index of the first occurrence of the pattern string inside the text string. If the pattern string doesn't appear inside the text string, return an appropriate sentinel value. Although virtually every programming language has some library function that does this (e.g. `strstr`, `string::find`, `String.indexOf`, etc.), we'd like you to implement this function without using those functions. Assuming the text string has length m and the pattern string has length n, what is the time complexity of your solution (O($m + n$)? O($m \log n$)? O(mn)? O(m^n)?)

Problem Four: Binary Search

In a programming language of your choice, write an implementation of binary search. Your implementation should return the index of the key in the array if it exists and an appropriate sentinel otherwise. Make sure your implementation runs in time O(log n). (Again, many programming languages have an implementation of this function in the standard libraries – `bsearch`, `std::lower_bound`, `Arrays.binarySearch`, etc. – but please refrain from using them.)

Problem Five: Know Your Tools!

Below is a list of common algorithms, data structures, and operations on those data structures. For each data structure, give the name of the implementation of that data structure in your main programming language, then list the names of the methods or functions that implement the specified operations. For each algorithm, figure out what standard library function or method implements that algorithm and briefly show off some code using it.

Not all languages have built-in libraries that perform all of these operations – C is notably lacking in this department, for example – and if that's the case, don't worry about it. However, be sure that you've looked carefully before concluding that there is no standard library that does what you want. Many languages have weird names for common data structures and operations.

- String
    - Search, Reverse
- Dynamic array
    - Append, Insert, Delete
- Hash table
    - Insert, Delete, Contains-Key, Lookup
- Linked List
    - Append, Prepend, Insert, Delete, Split, Splice
- Stack
    - Push, Pop, Is Empty
- Queue
    - Enqueue, Dequeue, Is Empty
- Priority Queue
    - Enqueue, Dequeue, Is Empty
- Balanced binary search tree
    - Insert, Delete, Contains-Key, Lookup
- Binary search
- Some fast sorting algorithm (quicksort, heapsort, etc.)

Problem Six: Data Structures

Implement a dynamic array (similar to `std::vector` or `ArrayList`) in a language of your choice. Feel free to have your dynamic array only store elements of some fixed type if that makes your life a bit easier. What is the big-O time complexity of each of the operations?

(more space for Problem Six)

Problem Seven: Regular Expressions

Write a function that takes as input the name of a file and returns a list of all the US phone numbers in that file. Make reasonable guesses for what the format of those phone numbers will be. Your function should have appropriate behavior in the event that the file cannot be opened.

Problem Eight: Binary Representations

Write a function in a programming language of your choice that accepts as input a non-negative integer, then reports how many 1's are in the binary representation of that number.

Problem Nine: Tree Structures

Write a function in a programming language of your choice that accepts as input a pointer to the root of a binary search tree containing integers, then returns the sum of all the numbers in that binary search tree.

Problem Ten: Object Orientation*

1. What's the difference between a class and an object?

2. What is meant by the term “encapsulation?” What language features support it?

3. What is a constructor? What is a destructor (or finalizer)? What should they do?

4. What is a static method? How is it different from a nonstatic method?

5. What is overriding? How does a language of your choice support overriding?

6. What is an abstract class? How do you declare an abstract class?

7. What is an interface? How does it differ from a class?

---

* Based on a list of questions from https://sites.google.com/site/steveyegge2/five-essential-phone-screen-questions.