Problem One: Fizz Bazz Buzz

Fizz bazz buzz is a game for a group of people sitting in a circle. The group counts upward from 0, saying the numbers in ascending order. However, there are some restrictions – any time a person would say a number that's a multiple of three, they instead say “fizz.” Any time a person would say a number that's multiple of five, they instead say “bazz.” Finally, any time a person would say a number that's a multiple of three *and* a multiple of five, they instead say “buzz.” (0 counts as a multiple of every number, by the way.)

For example, the first rounds of fizz bazz buzz would be

```
Buzz
  1
  2
Fizz
  4
Bazz
Fizz
  7
  8
Fizz
Bazz
 11
Fizz
 13
 14
Buzz
```

Write a program that prompts the user for a number, then plays that many rounds of fizz bazz buzz. For example, if the user typed in 15, you should see the above output.

Problem Two: Array Processing

Write a function that accepts as input a 2D array, then returns a new 2D array formed by rotating the original array 90°. For example, given the input

```
 1  2  3  4
 5  6  7  8
 9 10 11 12
```

your function should produce this output:

```
 9  5  1
10  6  2
11  7  3
12  8  4
```

Problem Three: File Processing

Write a function that takes as input the name of a file containing a list of integers, then returns an array containing all of the distinct integers in that file, with no duplicates, sorted into ascending order. For example, given this file:

```
3
3
7
1
7
1
7
1
```

Your function should return the array `[1, 3, 7]`. Then, analyze the time complexity of your solution (O(log n)? O(n)? O(n log n)? O(n^2)?)

Problem Four: Linked Lists

Write a function in a programming language of your choice that accepts as input a singly-linked list and a number k, then returns the kth-to-last element of the singly-linked list. If there aren't k elements in the list, your function should report an error through an appropriate mechanism. Please define your own linked list types; do not use the built-in linked list type in your language (e.g. `std::list`, `LinkedList`, etc.). Then, analyze the time complexity of your solution.

Problem Five: String Algorithms

Write a function that takes as input a sentence represented as a string, then reverses the order of the words in that sentence. Then, analyze the time complexity of your solution.

If you know what space complexity is, as a challenge, see if you can solve this with O(1) space complexity.

Problem Six: Binary Search

In a programming language of your choice, write an implementation of binary search. If there are multiple copies of the same key in the array, return the index of the first of them. Make sure your solution runs in time O(log n).

Problem Seven: Know Your Tools!

Search online for answers to the following questions for the *two* languages you are most comfortable using.

1. What is the best reference book or website for that language? Don't just go with the first Google hit – look around and see what the experts think.

2. What specific concepts in that programming language are often discussed in job interviews to assess whether you're familiar with that language? If you don't already know what these language features are, list two or three resources you could use to learn more about them.

3. Most programming languages undergo major revisions from time to time. What is the most recent revision of your programming language of choice? What are some of the changes that were introduced?

Problem Eight: Data Structures

Implement a hash table in a language of your choice. If that language has a standard set of hash functions, use one of those hash functions. Feel free to have your hash table only store keys and values of some fixed types if that makes your life a bit easier.

(more space for Problem Eight)

Problem Nine: Regular Expressions

Write a regular expression that matches amounts of currencies. You should be able to read values like the following:

- $137
- $9.99
- $12,345,678.90
- €2,34
- €10.000,00

For a fun challenge, look online for a list of common currencies and see if you can build a regular expression that matches as many of them as possible. Be aware of the conventions involving commas and dots, which change overseas.

Problem Ten: Binary Representations

Write a function that accepts as input an integer k, then returns the integer formed by reversing the order of the bits in k.

Problem Eleven: Tree Structures

Read up on each of the following data structures. Briefly describe where each of them would be most useful.

1. Binary search tree.

2. Trie.

3. Binary heap.

4. B-tree.

Problem Twelve: Object Orientation

Look up the following terms. Briefly define them and give an example of where they'd be useful.

1. Copy-on-write.

2. Factory method.

3. Singleton.

4. Covariance.

5. Contravariance.

6. Generic / Template