**General**

1. Find the most frequent integer in an array
2. Find pairs in an integer array whose sum is equal to 10 (bonus: do it in linear time)
3. Write fibbonaci iteratively and recursively (bonus: use dynamic programming)
4. Find the elements in an array that only occurs once.
5. Find the common elements of 2 int arrays
6. Implement binary search of a sorted array of integers
7. Find the first X prime numbers
8. Write a function that prints out the binary form of an int
9. Implement parseInt
10. Implement squareroot function
11. Implement an exponent function (bonus: now try in log(n) time)
12. Write a multiply function that multiples 2 integers without using \*
13. HARD: Given a function rand5() that returns a random int between 0 and 5, implement rand7()
14. HARD: Given a 2D array of 1s and 0s, count the number of "islands of 1s" (e.g. groups of connecting 1s)
15. Transpose a matrix of integers
16. Rotate a matrix of integers 90 degrees
17. Slide values in a matrix an arbitrary amount up/down and left/right

**Strings**

1. Find the first non-repeated character in a String
2. Reverse a String iteratively and recursively
3. Determine if 2 Strings are anagrams
4. Check if String is a palindrome
5. Check if a String is composed of all unique characters
6. Determine if a String is an int or a double
7. HARD: Find the shortest palindrome in a String
8. HARD: Print all permutations of a String
9. HARD: Given a single-line text String and a maximum width value, write the function 'String justify(String text, int maxWidth)' that formats the input text using full-justification, i.e., extra spaces on each line are equally distributed between the words; the first word on each line is flushed left and the last word on each line is flushed right
10. Ladder of strings, given dictionary of 5000, 5 letter strings

**Trees**

1. Implement a BST with insert and delete functions
2. Print a tree using BFS and DFS
3. Write a function that determines if a tree is a BST
4. Find the smallest element in a BST
5. Find the 2nd largest number in a BST
6. Given a binary tree which is a sum tree (child nodes add to parent), write an algorithm to determine whether the tree is a valid sum tree
7. Find the distance between 2 nodes in a BST and a normal binary tree
8. Print the coordinates of every node in a binary tree, where root is 0,0
9. Print a tree by levels
10. Given a binary tree which is a sum tree, write an algorithm to determine whether the tree is a valid sum tree
11. Given a tree, verify that it contains a subtree.
12. HARD: Find the max distance between 2 nodes in a BST.
13. HARD: Construct a BST given the pre-order and in-order traversal Strings

**Stacks, Queues, and Heaps**

1. Implement a stack with push and pop functions
2. Implement a queue with queue and dequeue functions
3. Find the minimum element in a stack in O(1) time
4. Write a function that sorts a stack (bonus: sort the stack in place without extra memory)
5. Implement a binary min heap. Turn it into a binary max heap
6. HARD: Implement a queue using 2 stacks

**Linked Lists**

1. Implement a linked list (with insert and delete functions)
2. Find the Nth element in a linked list
3. Remove the Nth element of a linked list
4. Check if a linked list has cycles
5. Given a circular linked list, find the node at the beginning of the loop. Example: A-->B-->C --> D-->E -->C, C is the node that begins the loop
6. Check whether a link list is a palindrome
7. Reverse a linked list iteratively and recursively

**Sorting**

1. Implement bubble sort
2. Implement selection sort
3. Implement insertion sort
4. Implement merge sort
5. Implement quick sort