**General**

* Find the most frequent integer in an array
* Find pairs in an integer array whose sum is equal to 10 (bonus: do it in linear time)
* Given 2 integer arrays, determine if the 2nd array is a rotated version of the 1st array. Ex. Original Array A={1,2,3,5,6,7,8} Rotated Array B={5,6,7,8,1,2,3}
* Write fibbonaci iteratively and recursively (bonus: use dynamic programming)
* Find the only element in an array that only occurs once.
* Find the common elements of 2 int arrays
* Implement binary search of a sorted array of integers
* Implement binary search in a rotated array (ex. {5,6,7,8,1,2,3})
* Use dynamic programming to find the first X prime numbers
* Write a function that prints out the binary form of an int
* Implement parseInt
* Implement squareroot function
* Implement an exponent function (bonus: now try in log(n) time)
* Write a multiply function that multiples 2 integers without using \*
* HARD: Given a function rand5() that returns a random int between 0 and 5, implement rand7()
* HARD: Given a 2D array of 1s and 0s, count the number of "islands of 1s" (e.g. groups of connecting 1s)

**Strings**

* Find the first non-repeated character in a String
* Reverse a String iteratively and recursively
* Determine if 2 Strings are anagrams
* Check if String is a palindrome
* Check if a String is composed of all unique characters
* Determine if a String is an int or a double
* HARD: Find the shortest palindrome in a String
* HARD: Print all permutations of a String
* 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

**Trees**

* Implement a BST with insert and delete functions
* Print a tree using BFS and DFS
* Write a function that determines if a tree is a BST
* Find the smallest element in a BST
* Find the 2nd largest number in a BST
* 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
* Find the distance between 2 nodes in a BST and a normal binary tree
* Print the coordinates of every node in a binary tree, where root is 0,0
* Print a tree by levels
* Given a binary tree which is a sum tree, write an algorithm to determine whether the tree is a valid sum tree
* Given a tree, verify that it contains a subtree.
* HARD: Find the max distance between 2 nodes in a BST.
* HARD: Construct a BST given the pre-order and in-order traversal Strings

**Stacks, Queues, and Heaps**

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

**Linked Lists**

* Implement a linked list (with insert and delete functions)
* Find the Nth element in a linked list
* Remove the Nth element of a linked list
* Check if a linked list has cycles
* 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
* Check whether a link list is a palindrome
* Reverse a linked list iteratively and recursively

**Sorting**

* Implement bubble sort
* Implement selection sort
* Implement insertion sort
* Implement merge sort
* Implement quick sort