Format: All *questions* first, then all *answers* (Python + JavaScript for each).

Connecteam Coding Practice — Questions

1. **1. Two Sum**

   Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target. Assume exactly one solution, no reuse of element.

2. **2. Contains Duplicate**

   Given an integer array nums, return true if any value appears at least twice; otherwise false.

3. **3. Valid Anagram**

   Given two strings s and t, return true if t is an anagram of s.

4. **4. First Unique Character**

   Given a string s, find the index of the first non-repeating character. If none, return -1.

5. **5. Move Zeroes**

   Move all 0s to the end of the array while maintaining the relative order of non-zero elements. Modify in-place.

6. **6. Best Time to Buy and Sell Stock**

   Given prices[i] is the price on day i, return the max profit you can achieve from one transaction.

7. **7. Maximum Subarray (Kadane)**

   Return the largest sum of a contiguous subarray.

8. **8. Merge Two Sorted Arrays**

   Given two sorted arrays, return a merged sorted array.

9. **9. Plus One**

   Given a non-empty array of digits representing a non-negative integer, add one to the integer.

10. **10. Rotate Array**

Rotate an array to the right by k steps (k may be greater than length).

11. **11. Intersection of Two Arrays (Unique)**

    Return the unique intersection of two arrays.

12. **12. Group Anagrams**

    Group a list of strings by anagram equivalence, return list of groups.

13. **13. Top K Frequent Elements**

    Return the k most frequent elements from an integer array.

14. **14. Product of Array Except Self**

    Return an array answer where answer[i] is the product of all the elements of nums except nums[i]. No division.

15. **15. Valid Parentheses**

    Given a string of brackets '()[]{}', determine if it's valid (properly closed/ordered).

16. **16. Longest Common Prefix**

    Find the longest common prefix among an array of strings.

17. **17. Single Number (XOR)**

    Given a non-empty array where every element appears twice except for one, find that single one.

18. **18. Majority Element**

    Find the element that appears more than ⌊n/2⌋ times. Assume it exists.

19. **19. Missing Number**

    Given an array containing n distinct numbers in the range [0, n], return the one that is missing.

20. **20. Pascal's Triangle Row**

    Return the k-th (0-indexed) row of Pascal's Triangle.

21. **21. Roman to Integer**

Convert a Roman numeral to integer.

22. **22. Excel Column Number**

    Given a string columnTitle that represents the column title as appear in an Excel sheet, return its corresponding column number.

23. **23. Happy Number**

    Write an algorithm to determine if a number n is happy.

24. **24. Subsets (Power Set)**

    Given a set of distinct integers, return all possible subsets (the power set).

25. **25. Permutations**

    Given a set of distinct integers, return all possible permutations.

26. **26. Climbing Stairs**

    You are climbing a staircase. It takes n steps to reach the top. Each time you can climb 1 or 2 steps. In how many distinct ways can you climb to the top?

27. **27. Two Sum II (Input Array Is Sorted)**

    Return 1-indexed indices of two numbers such that they add up to target in a sorted array.

28. **28. Isomorphic Strings**

    Given two strings s and t, determine if they are isomorphic (characters in s can be replaced to get t).

29. **29. Ransom Note**

    Given two strings ransomNote and magazine, return true if ransomNote can be constructed from magazine.

30. **30. Valid Sudoku**

    Determine if a 9x9 Sudoku board is valid (no duplicates in any row, column, or 3x3 box).

Answers — Python and JavaScript

1. **1. Two Sum**

Python

```
def two_sum(nums, target):
    seen = {}
    for i, x in enumerate(nums):
        if target - x in seen:
            return [seen[target - x], i]
        seen[x] = i
```

JavaScript

```
function twoSum(nums, target) {
  const seen = new Map();
  for (let i = 0; i < nums.length; i++) {
    const x = nums[i];
    if (seen.has(target - x)) return [seen.get(target - x), i];
    seen.set(x, i);
  }
}
```

2. ## 2. Contains Duplicate

Python

```
def contains_duplicate(nums):
    return len(set(nums)) != len(nums)
```

JavaScript

```
function containsDuplicate(nums) {
  return new Set(nums).size !== nums.length;
}
```

3. ## 3. Valid Anagram

Python

```python
def is_anagram(s, t):
    if len(s) != len(t): return False
    from collections import Counter
    return Counter(s) == Counter(t)
```

JavaScript

```javascript
function isAnagram(s, t) {
  if (s.length !== t.length) return false;
  const count = new Map();
  for (const ch of s) count.set(ch, (count.get(ch) || 0) + 1);
  for (const ch of t) {
    if (!count.has(ch)) return false;
    count.set(ch, count.get(ch) - 1);
    if (count.get(ch) === 0) count.delete(ch);
  }
  return count.size === 0;
}
```

4. ## 4. First Unique Character

Python

```python
def first_uniq_char(s):
    from collections import Counter
    freq = Counter(s)
    for i, ch in enumerate(s):
        if freq[ch] == 1:
            return i
    return -1
```

JavaScript

```javascript
function firstUniqChar(s) {
  const freq = new Map();
  for (const ch of s) freq.set(ch, (freq.get(ch) || 0) + 1);
  for (let i = 0; i < s.length; i++) if (freq.get(s[i]) === 1) return i;
```

```
  return -1;
}
```

5. ## 5. Move Zeroes

Python

```
def move_zeroes(nums):
    j = 0
    for i in range(len(nums)):
        if nums[i] != 0:
            nums[j], nums[i] = nums[i], nums[j]
            j += 1
```

JavaScript

```
function moveZeroes(nums) {
  let j = 0;
  for (let i = 0; i < nums.length; i++) {
    if (nums[i] !== 0) {
      [nums[j], nums[i]] = [nums[i], nums[j]];
      j++;
    }
  }
}
```

6. ## 6. Best Time to Buy and Sell Stock

Python

```
def max_profit(prices):
    minp = float('inf'); ans = 0
    for p in prices:
        minp = min(minp, p)
        ans = max(ans, p - minp)
    return ans
```

JavaScript

```
function maxProfit(prices) {
  let minp = Infinity, ans = 0;
  for (const p of prices) {
    minp = Math.min(minp, p);
    ans = Math.max(ans, p - minp);
  }
  return ans;
}
```

7. 7. Maximum Subarray (Kadane)

Python

```
def max_sub_array(nums):
    best = cur = nums[0]
    for x in nums[1:]:
        cur = max(x, cur + x)
        best = max(best, cur)
    return best
```

JavaScript

```
function maxSubArray(nums) {
  let best = nums[0], cur = nums[0];
  for (let i = 1; i < nums.length; i++) {
    cur = Math.max(nums[i], cur + nums[i]);
    best = Math.max(best, cur);
  }
  return best;
}
```

8. 8. Merge Two Sorted Arrays

Python

```
def merge_sorted(a, b):
    i=j=0; res=[]
    while i < len(a) and j < len(b):
```

```
        if a[i] <= b[j]: res.append(a[i]); i+=1
        else: res.append(b[j]); j+=1
    res.extend(a[i:]); res.extend(b[j:])
    return res
```

JavaScript

```
function mergeSorted(a, b) {
  let i = 0, j = 0, res = [];
  while (i < a.length && j < b.length) {
    if (a[i] <= b[j]) res.push(a[i++]); else res.push(b[j++]);
  }
  return res.concat(a.slice(i)).concat(b.slice(j));
}
```

9. ## 9. Plus One

Python

```
def plus_one(d):
    i = len(d) - 1
    while i >= 0 and d[i] == 9:
        d[i] = 0
        i -= 1
    if i >= 0:
        d[i] += 1
        return d
    return [1] + d
```

JavaScript

```
function plusOne(d) {
  let i = d.length - 1;
  while (i >= 0 && d[i] === 9) { d[i] = 0; i--; }
  if (i >= 0) { d[i] += 1; return d; }
  return [1, ...d];
}
```

10. ## 10. Rotate Array

Python

```python
def rotate(nums, k):
    n = len(nums); k %= n
    def rev(l, r):
        while l < r:
            nums[l], nums[r] = nums[r], nums[l]
            l += 1; r -= 1
    rev(0, n-1); rev(0, k-1); rev(k, n-1)
```

JavaScript

```javascript
function rotate(nums, k) {
  const n = nums.length; k %= n;
  const rev = (l, r) => { while (l < r) { [nums[l], nums[r]] = [nums[r], nums
  rev(0, n-1); rev(0, k-1); rev(k, n-1);
}
```

11. ## 11. Intersection of Two Arrays (Unique)

Python

```python
def intersection(a, b):
    return list(set(a) & set(b))
```

JavaScript

```javascript
function intersection(a, b) {
  const A = new Set(a), B = new Set(b), res = [];
  for (const x of A) if (B.has(x)) res.push(x);
  return res;
}
```

12. ## 12. Group Anagrams

Python

```python
def group_anagrams(strs):
    from collections import defaultdict
    groups = defaultdict(list)
    for s in strs:
        key = tuple(sorted(s))
        groups[key].append(s)
    return list(groups.values())
```

JavaScript

```javascript
function groupAnagrams(strs) {
  const groups = new Map();
  for (const s of strs) {
    const key = s.split('').sort().join('');
    if (!groups.has(key)) groups.set(key, []);
    groups.get(key).push(s);
  }
  return Array.from(groups.values());
}
```

13. ## 13. Top K Frequent Elements

Python

```python
def top_k_frequent(nums, k):
    from collections import Counter
    return [x for x, _ in Counter(nums).most_common(k)]
```

JavaScript

```javascript
function topKFrequent(nums, k) {
  const freq = new Map();
  for (const x of nums) freq.set(x, (freq.get(x) || 0) + 1);
  return Array.from(freq.entries()).sort((a,b)=>b[1]-a[1]).slice(0,k).map(e=>
}
```

14. ## 14. Product of Array Except Self

Python

```python
def product_except_self(nums):
    n = len(nums)
    res = [1]*n
    prefix = 1
    for i in range(n):
        res[i] = prefix
        prefix *= nums[i]
    suffix = 1
    for i in range(n-1, -1, -1):
        res[i] *= suffix
        suffix *= nums[i]
    return res
```

JavaScript

```javascript
function productExceptSelf(nums) {
  const n = nums.length;
  const res = Array(n).fill(1);
  let prefix = 1;
  for (let i = 0; i < n; i++) { res[i] = prefix; prefix *= nums[i]; }
  let suffix = 1;
  for (let i = n-1; i >= 0; i--) { res[i] *= suffix; suffix *= nums[i]; }
  return res;
}
```

15. 15. Valid Parentheses

Python

```python
def is_valid(s):
    stack = []
    pair = {')':'(', ']':'[', '}':'{'}
    for ch in s:
        if ch in '([{':
            stack.append(ch)
        else:
            if not stack or stack[-1] != pair.get(ch):
                return False
```

```
            stack.pop()
    return not stack
```

JavaScript

```
function isValid(s) {
  const stack = [], pair = {')':'(', ']':'[', '}':'{'};
  for (const ch of s) {
    if (ch === '(' || ch === '[' || ch === '{') stack.push(ch);
    else {
      if (!stack.length || stack[stack.length-1] !== pair[ch]) return false;
      stack.pop();
    }
  }
  return stack.length === 0;
}
```

16. ## 16. Longest Common Prefix

Python

```
def longest_common_prefix(strs):
    if not strs: return ""
    prefix = strs[0]
    for s in strs[1:]:
        while not s.startswith(prefix):
            prefix = prefix[:-1]
            if not prefix: return ""
    return prefix
```

JavaScript

```
function longestCommonPrefix(strs) {
  if (!strs.length) return "";
  let prefix = strs[0];
  for (let i = 1; i < strs.length; i++) {
    while (!strs[i].startsWith(prefix)) {
      prefix = prefix.slice(0, -1);
      if (!prefix) return "";
```

```
    }
  }
  return prefix;
}
```

17. ## 17. Single Number (XOR)

Python

```
def single_number(nums):
    x = 0
    for n in nums: x ^= n
    return x
```

JavaScript

```
function singleNumber(nums) {
  let x = 0;
  for (const n of nums) x ^= n;
  return x;
}
```

18. ## 18. Majority Element

Python

```
def majority_element(nums):
    count = 0; cand = None
    for x in nums:
        if count == 0: cand = x
        count += 1 if x == cand else -1
    return cand
```

JavaScript

```
function majorityElement(nums) {
  let count = 0, cand = null;
```

```
  for (const x of nums) {
    if (count === 0) cand = x;
    count += (x === cand) ? 1 : -1;
  }
  return cand;
}
```

19. ## 19. Missing Number

Python

```
def missing_number(nums):
    n = len(nums)
    return n*(n+1)//2 - sum(nums)
```

JavaScript

```
function missingNumber(nums) {
  const n = nums.length;
  return (n*(n+1))/2 - nums.reduce((a,b)=>a+b,0);
}
```

20. ## 20. Pascal's Triangle Row

Python

```
def get_row(k):
    row = [1]
    for _ in range(k):
        row = [1] + [row[i]+row[i+1] for i in range(len(row)-1)] + [1]
    return row
```

JavaScript

```
function getRow(k) {
  let row = [1];
  for (let r = 0; r < k; r++) {
```

```
    const next = [1];
    for (let i = 0; i < row.length - 1; i++) next.push(row[i] + row[i+1]);
    next.push(1);
    row = next;
  }
  return row;
}
```

21. 21. Roman to Integer

Python

```
def roman_to_int(s):
    vals = {'I':1,'V':5,'X':10,'L':50,'C':100,'D':200//2,'M':1000}
    # Correct D should be 500; keep correct mapping:
    vals = {'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000}
    ans = 0
    for i, ch in enumerate(s):
        v = vals[ch]
        if i+1 < len(s) and v < vals[s[i+1]]:
            ans -= v
        else:
            ans += v
    return ans
```

JavaScript

```
function romanToInt(s) {
  const vals = {I:1,V:5,X:10,L:50,C:100,D:500,M:1000};
  let ans = 0;
  for (let i = 0; i < s.length; i++) {
    const v = vals[s[i]];
    if (i+1 < s.length && v < vals[s[i+1]]) ans -= v; else ans += v;
  }
  return ans;
}
```

22. 22. Excel Column Number

Python

```
def title_to_number(s):
    ans = 0
    for ch in s:
        ans = ans*26 + (ord(ch)-ord('A')+1)
    return ans
```

JavaScript

```
function titleToNumber(s) {
  let ans = 0;
  for (const ch of s) ans = ans*26 + (ch.charCodeAt(0)-'A'.charCodeAt(0)+1);
  return ans;
}
```

23. 23. Happy Number

Python

```
def is_happy(n):
    def next_num(x):
        s = 0
        while x:
            x, d = divmod(x, 10)
            s += d*d
        return s
    seen = set()
    while n != 1 and n not in seen:
        seen.add(n)
        n = next_num(n)
    return n == 1
```

JavaScript

```
function isHappy(n) {
  const nextNum = (x) => {
    let s = 0;
    while (x) { const d = x % 10; s += d*d; x = Math.floor(x/10); }
```

```
    return s;
  };
  const seen = new Set();
  while (n !== 1 && !seen.has(n)) { seen.add(n); n = nextNum(n); }
  return n === 1;
}
```

24. 24. Subsets (Power Set)

Python

```
def subsets(nums):
    res = [[]]
    for x in nums:
        res += [r + [x] for r in res]
    return res
```

JavaScript

```
function subsets(nums) {
  let res = [[]];
  for (const x of nums) res = res.concat(res.map(r => r.concat([x])));
  return res;
}
```

25. 25. Permutations

Python

```
def permute(nums):
    res = []
    def backtrack(start=0):
        if start == len(nums):
            res.append(nums[:]); return
        for i in range(start, len(nums)):
            nums[start], nums[i] = nums[i], nums[start]
            backtrack(start+1)
            nums[start], nums[i] = nums[i], nums[start]
```

```
    backtrack()
    return res
```

JavaScript

```
function permute(nums) {
  const res = [];
  const backtrack = (start=0) => {
    if (start === nums.length) { res.push(nums.slice()); return; }
    for (let i = start; i < nums.length; i++) {
      [nums[start], nums[i]] = [nums[i], nums[start]];
      backtrack(start+1);
      [nums[start], nums[i]] = [nums[i], nums[start]];
    }
  };
  backtrack();
  return res;
}
```

26. 26. Climbing Stairs

Python

```
def climb_stairs(n):
    a, b = 1, 1
    for _ in range(n):
        a, b = b, a+b
    return a
```

JavaScript

```
function climbStairs(n) {
  let a = 1, b = 1;
  for (let i = 0; i < n; i++) { [a, b] = [b, a+b]; }
  return a;
}
```

27. 27. Two Sum II (Input Array Is Sorted)

Python

```
def two_sum_sorted(nums, target):
    l, r = 0, len(nums)-1
    while l < r:
        s = nums[l] + nums[r]
        if s == target: return [l+1, r+1]
        if s < target: l += 1
        else: r -= 1
```

JavaScript

```
function twoSumSorted(nums, target) {
  let l = 0, r = nums.length - 1;
  while (l < r) {
    const s = nums[l] + nums[r];
    if (s === target) return [l+1, r+1];
    if (s < target) l++; else r--;
  }
}
```

28. ## 28. Isomorphic Strings

Python

```
def is_isomorphic(s, t):
    if len(s) != len(t): return False
    m1 = {}; m2 = {}
    for a, b in zip(s, t):
        if (a in m1 and m1[a] != b) or (b in m2 and m2[b] != a):
            return False
        m1[a] = b; m2[b] = a
    return True
```

JavaScript

```
function isIsomorphic(s, t) {
  if (s.length !== t.length) return false;
```

```
  const m1 = new Map(), m2 = new Map();
  for (let i = 0; i < s.length; i++) {
    const a = s[i], b = t[i];
    if ((m1.has(a) && m1.get(a) !== b) || (m2.has(b) && m2.get(b) !== a)) ret
    m1.set(a, b); m2.set(b, a);
  }
  return true;
}
```

29. ## 29. Ransom Note

Python

```
def can_construct(ransomNote, magazine):
    from collections import Counter
    c1, c2 = Counter(ransomNote), Counter(magazine)
    for ch, cnt in c1.items():
        if c2[ch] < cnt: return False
    return True
```

JavaScript

```
function canConstruct(ransomNote, magazine) {
  const c1 = new Map(), c2 = new Map();
  for (const ch of ransomNote) c1.set(ch, (c1.get(ch)||0)+1);
  for (const ch of magazine) c2.set(ch, (c2.get(ch)||0)+1);
  for (const [ch, cnt] of c1.entries()) if ((c2.get(ch)||0) < cnt) return fal
  return true;
}
```

30. ## 30. Valid Sudoku

Python

```
def is_valid_sudoku(board):
    rows = [set() for _ in range(9)]
    cols = [set() for _ in range(9)]
    boxes = [set() for _ in range(9)]
    for r in range(9):
```

```
        for c in range(9):
            v = board[r][c]
            if v == '.': continue
            b = (r//3)*3 + (c//3)
            if v in rows[r] or v in cols[c] or v in boxes[b]:
                return False
            rows[r].add(v); cols[c].add(v); boxes[b].add(v)
    return True
```

JavaScript

```
function isValidSudoku(board) {
  const rows = Array.from({length:9}, ()=>new Set());
  const cols = Array.from({length:9}, ()=>new Set());
  const boxes = Array.from({length:9}, ()=>new Set());
  for (let r = 0; r < 9; r++) {
    for (let c = 0; c < 9; c++) {
      const v = board[r][c];
      if (v === '.') continue;
      const b = Math.floor(r/3)*3 + Math.floor(c/3);
      if (rows[r].has(v) || cols[c].has(v) || boxes[b].has(v)) return false;
      rows[r].add(v); cols[c].add(v); boxes[b].add(v);
    }
  }
  return true;
}
```