Coding Round Question Set

1. **Bracket Validator**

   You are designing a compiler for a C++ program and need to check that braces in any given file are balanced. Braces in a string are considered to be balanced if the following criteria are met: All braces must be closed. Braces come in pairs of the form (), {} and []. The left brace opens the pair, and the right one closes it. In any set of nested braces, the braces between any pair must be closed. For example, [{}] is a valid grouping of braces but [}]{} is not.

   Function Description
   Write the function **braces**. The function must return take input S as String & return true if String is valid brace expression.

   Constraints:
   It is guaranteed that each value in String consists of (,), {, }, [, and] only.

2. **Using Functions**

   A compliance system monitors incoming and outbound calls and sends an alert whenever the average calls over a trailing number of minutes exceeds a threshold. If trailing minutes to consider, preceedingMinutes = 5, at time T, average the call volumes for times T-(5-1), T-(5-2) ...T.

   For example, the calls over the last n = 8 minutes are represented in the array numCalls = [2, 2, 2, 2, 5, 5, 5, 8]. The threshold, alertThreshold = 4 and the trailing values to consider, preceedingMinutes = 3. No alerts will be sent until at least T = 3 because there are not enough values to consider. At T = 3, average calls = (2 + 2 + 2)/3 = 2. Average calls over the windows from T = 3 to the end at T = 8 are 2, 2, 3, 4, 5, and 6. A total of 2 alerts are sent during the last two periods. Given data as described, determine the number of alerts sent by the end of the timeframe.

   Function Description
   Write the **numberOfAlerts**. It should return an integer that represents the number of alerts sent over the timeframe.

   numberOfAlerts has the following parameter(s):
   - **preceedingMinutes**: an integer that represents the trailing number of minutes to consider
   - **alertThreshold**: an integer that represents the maximum number of calls allowed without triggering an alert
   - **numCalls[numCalls[0]..numCalls[n-1]]:** an array of integers where each numCalls[i] represents the number of calls made during minute i

3. **Palindrome Algorithm**

4. **Anagram Algo and variations**

   Based on anagram's logic, display the output in the below format.

   Input: ['cat', 'dog', 'cat', 'god'] #list of words
   Output: [{'cat'} , {'dog','god'}] #list of sets

5. **Binary Search Algorithm Implementation**

6. **Strings 1**

   Write a java program to generate the sequences based on the length passed as an input, from the given input string.
   For example, if the input string is "abc", and input length is 2, then the output must be {ab,bc} only (only the consecutive sequences allowed)
   If the input is "abcde" and length is 3, then output must be {abc, bcd, cde}

7. **Strings 2**

   A magic potion can be made out of the formula when added in a particular order. The formula used can be from {A,B,C,D,E,F,G,H} and in any order
   For example, if the formula is made with the following sequence ABCDABCE , then the number of steps to achieve that formula would be 8.
   If the input is ABCABCE, then we can place * for the repeating sequence, ABC*E and the number of steps to achieve the formula would be 5.
   Write a program for achieving the same

8. **Second Smallest element in an array**

9. **First Non–Duplicate character in string**

   Write a java program to find the first non-duplicate character in the string.
   For example, if the input is apple, then the output would be 'a'
   If the input is racecars, then the output would be 'e'

10. **String Manipulation**

    You are merging data from two sources connected to a network access point to create a new data packet.
    You must merge strings a and b, and then return a single merged string. A merge operation on two strings is described as follows:
    Append alternating characters from a and b, respectively, to some new string, mergedString.
    Once all of the characters in one of the strings have been merged, append the remaining characters in the other string to mergedString.

    As an example, assume you have two strings to merge: 'abc' and 'stuvwx'. Alternate between the first and second strings as long as you can:

'a' + 's' + 'b' + 't' + 'c' + 'u'. At this point you have traversed the first string and have generated 'asbtcu'. The remainder of the second string, 'vwx' is now added to the end of the string, creating 'asbtcuvwx'.

Function Description
Write the function mergeStrings. The function must return the merged string.

mergeStrings has the following parameter(s):
a: first string
b: second string

11. Binary Tree Searching Algorithm and its implementation

12. Max Difference

You are given an array of integers and must compute the maximum difference between any item and any lower indexed smaller item for all the possible pairs, i.e., for a given array a find the maximum value of a[j] - a[i] for all i, j where $0 \leq i < j < n$ and $a[i] < a[j]$. If there are no lower indexed smaller items for all the items, then return -1.

For example, given an array [1, 2, 6, 4], you would first compare 2 to the elements to its left. 1 is smaller, so calculate the difference 2 - 1 = 1. 6 is bigger than 2 and 1, so calculate the differences 4 and 5. 4 is only bigger than 2 and 1, and the differences are 2 and 3. The largest difference was 6 - 1 = 5.

Function Description
Write the function maxDifference. The function must return an integer representing the maximum difference in a.

maxDifference has the following parameter(s):
a[a[0],a[1],...a[n-1]]: an array of integers

13. Min-Max

Write down two programs to find the max and min in a list of numbers in most efficient way. Find the time complexity for both of them

14. Prime Number

Find the whether a number is prime number (by writing program)
a. Using recursion
b. Without using recursion
Find time complexity for both of them.

15. Singly and doubly linked list implementation

Searching, inserting and deleting an element with their time complexities.

16. Class President

A group of students are sitting in a circle. The teacher is electing a new class president. The teacher does this by singing a song while walking around the circle. After the song is finished the student at which the teacher stopped is removed from the circle.
Starting at the student next to the one that was just removed, the teacher resumes singing and walking around the circle.
After the teacher is done singing, the next student is removed. The teacher repeats this until only one student is left.

A song of length k will result in the teacher walking past k students on each round. The students are numbered 1 to n. The teacher starts at student 1.

For example, suppose the song length is two (k=2). And there are four students to start with (1,2,3,4). The first
student to go would be `2`, after that `4`, and after that `3`. Student `1` would be the next president in this example.
Function Description:
Implement a function that takes n = the number of students sitting in a circle. , k= the length (in students) of each song and returns the number of the student that is elected. (between 1 to n)

17. Array Rotation

You have a sorted array of size N (N >= 2) which has been rotated x number of times (x > 0 and x < N).Array has distinct elements only. Now after rotation, array has 2 sorted parts. You have to find which part of array has greater sum.
Input: You will be given an array (Sorted array rotated x times).
Output: Implement a function to find which part of array has greater sum and return that sum.

18. Jumping Game

Alice and Bob are playing a jumping game. Initially Alice and Bob are standing at 0th tile in their respective rows of tiles. They will start jumping 'x' number of tiles in each step (Formally, step size for both will be same, 'x'). Alice has to reach tile number N and Bob has to reach tile number M. They want to take minimum number of jumps to reach their respective tiles (N and M). You have to find the jump size 'x' to minimize number of jumps taken by both given their final tile numbers N and M. They can't jump over final tile, for example if N is 10, then step size 'x' can't be 7 because Alice will never reach final tile number 10 then.
Input: You will be given two integers N and M.
Output: Implement a function to compute and return step size 'x'.

19. String-Reversal

Given an String. Reverse it without using inbuilt function.
Exp: ‘nirmal’ -- ‘lamrin’

20. **Arrays-1**

Find the loop and its length in jump Array. Given an array, where at each index, the value indicates next index to jump to.
Exp: [2,4,3,0,1] , at first we start from index 0 which have value 2 so we jump to index 2. At index 2 we have value 3 so we jump at index 0. So we started at index 0 -> 2 -> 3 -> 0 , this is loop. We need to detect these loops and length (here it is 4).

21. **Binary-Trees**

In a forest of binary tree, where each node has three pointers left pointer, right pointer and parent pointer(pointing to its parent). You have been given two nodes N1, N2. Find The immediate predecessor of these nodes.

Notes : they will make you write algorithm on paper and will ask about time and space complexity. All above three problems can be solved using O(n) time with O(1) space complexity.

22. **Find Max subset**

[1,-5,6,-10,9,-4,-7]

23. N length of array is given. All the elements are unique (No duplicate). You need to find out such element, that is lesser than its immediate right element as well as immediate left element and return that element, if found. For first element you should only check for its immediate right element and for last element you should only check for its immediate left element. If there is no such element, then return 0. Solve this in optimal way.

Example-

1) [10,8,7,-1,6,9]
Output: -1

2) [1,2,3,4,5,6,8]
Output: 1

24. Stack is given, that contains N elements, and all are unique (No Duplicate). We need to find out how many, minimum Push () and Pop () operations are required to sort that stack. Without using any extra space. Solve this in optimal way.
Example –
1) Stack = [1,10,3,15,2,0]
Output – 10 (5 – pop and 5 - push)
2) stack = [12,10,8,5,3,1]
Output – 0 (no push and no pop)

25. Four locks and four keys are given, need to find out key for respective lock. All are unique (No duplicate, means for one lock, one key). One lock_logic(lock,key) function is given that takes lock name and key name and return output as (0 : if lock and key matches, 1 : if key size is greater than lock size, -1 : if key size is lesser than lock size). Solve this in optimal way.

Note: Don't compare Lock name == Key Name (Example: A==A. That won't allowed)

Example – lock: [A,B,C,D]
Key: [D,C,A,B]
Output: Lock: [A,B,C,D]
Key: [A,B,C,D]

26. One array is given of size N. That contains only 0 or 1. You need to find out sub-array that have max() value by using below equations.

abs(Number of 0's - Number of 1's)

Example – [0,1,1,1,1,1,0]
Output - 5

27. Arrange coins alternatively as Heads and Tails with least number of steps taken for it. Given the inputs in format as *"H T H T T T"* for example.

28. Maximum possible valid time(as in 24hrs digital clock) with given 4 input digits. Sample input: "2 6 1 3". Valid time could be: a. 13:26, b. 21:36, etc.

29. A robot starts from a coordinate (0,0) and can move 1 unit at a time only in the following directions Up, Down, Left, Down denoted as {'U','D','L','R'} respectively.
A string of directions will be passed and the function should return the final coordinate the bot is in.
If the input string is having junk character, the bot should ignore it and remain still.
Example Input string ULLD Output: (-2,0)
Input string 'U R WELCOME ' Output: (0,1)

30. Find the best average grade of a student in a pool of class marks.
Given input as [['Bob',100],['Charles',30],['Bob',20],['Marie',80]]
Output:80
Asked to optimize the code if the list size is very high:

31. Was Asked the approach to solve the below problem:
The teacher needs to select the President among the students and its done through singing. The students are sitting in circles and they are numbered from 1. The teacher would start singing from student number 1 and move accordingly to 2,3,4 and so on. The teacher stops when the song ends. That student where song ends will be removed and the teacher will start singing from next student.
The process will continue till there is only one student left and will be declared as the President.
E.g. Input Group length and song length as [4,2]
So the students are [1,2,3,4] .
First student 2 will be removed, then 4, the 3 and student 1 will be remaining. Therefore president is student 1.

32. Given a IP information list find the most used IP. In input IP is followed by a a blank space.
Ex: input = ['10.10.10.1 BroXX yyyy ZZZ','10.10.10.1 BroXXd ZZZ','10.10.10.2 ddddd yyy ZZZ']

Output : 10.10.10.1 (repeated two times)

33. Adding two fractions and storing the result in a simplified form
Fraction = [fraction1, fraction2]
Fraction1 = [numerator, denominator]
For ex: fraction = [[2,3],[1,2]], add these fractions to get result [7,6]

34. Given a list and integer k, find the length of shortest sublist or subarray where sum of the numbers is atleast k. If no such sub list is found return -1.
Ex : a = [[1,2,3,4],6], return 2 as a result
B = [[1,2,3,4],12] returns -1

35. Find missing alphabets in the given sentence and return the missing characters in sorted string.
Ex : 'The fox jumps over the lazy dog', output = '', because it has all characters.

36. Given a IP information list find the most used IP. In input IP is followed by a a blank space.
Ex: input = ['10.10.10.1 BroXX yyyy ZZZ','10.10.10.1 BroXXd ZZZ','10.10.10.2 ddddd yyy ZZZ']
Output : 10.10.10.1 (repeated two times)
Was asked to implement if there were two or more IPs with same number of entries (=max entires) in the file.

37. Check if a number is a power of 10. if the number was a power of 10, return 1, else zero.
They wanted to check for decimals also (Ex. for 0.1 the output will be -1 for function will return True)

38: Find all possible unique substring from a given string
Example 1:
i/p string: 'abc'
Expected output: {'ab', 'bc'}

39.: From the given input find the longest words that can created which are also present in the given dictionary. (Was asked to write an optimized and scalable code)
Given Dictionary: {to, toe, toes, dogs, dogs, banana}
Example 1:
i/p: 'ote'
Expected output: ['toe']
Example 2:
i/p: 'dotesg'
Expected output: {'dogs', 'toes'}