**Global Trend Programming Profile Assessment Questions**

Welcome to the Global Trend Programming Profile assessment. This document outlines the requirements and guidelines for completing the programming assessment. Please read the instructions carefully before you begin.

## **Assessment Overview**

This assessment consists of 10 programming questions designed to evaluate your problem-solving and coding skills in Java. You are required to attempt all the questions within a 1-day time period. The assessment is timed, and your submissions will be evaluated based on correctness, efficiency, and coding best practices.

## **Instructions**

1. **Time Limit**: You have 24 hours to complete all 10 questions from the moment you start the assessment.
2. **Programming Language**: All solutions must be written in Java.
3. **Submission**:
   * **Word Document**: Create a Word document containing your code for each question along with screenshots of the output. Name the file Programming\_Assessment\_YourName.docx.
   * **GitHub Profile**: Create a GitHub profile if you don't already have one. Upload all your code files to a new repository named GlobalTrend\_Programming\_Assessment.
4. **Evaluation Criteria**:
   * **Correctness**: The solution must provide the correct output for all test cases.
   * **Efficiency**: The solution should be optimized for time and space complexity.
   * **Code Quality**: Follow coding best practices, including proper naming conventions, comments, and code organization.

**Questions:**

#### Problem Statement

#### Design and implement a data structure for a Least Recently Used (LRU) cache. It should support the following operations: get and put.

get(key): Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.

put(key, value): Set or insert the value if the key is not already present. When the cache reaches its capacity, it should invalidate the least recently used item before inserting a new item.

#### Constraints

#### The number of get and put operations will be in the range [1, 10^5].

The capacity of the cache is between 1 and 10^5.

1. Write a Java program that demonstrates the ConcurrentModificationException. Explain why the exception is thrown and how to handle it properly.
2. Create a custom annotation @LogExecutionTime to log the execution time of annotated methods. Implement an annotation processor to handle this annotation.
3. Problem Statement

Design an algorithm to serialize and deserialize a binary tree. Implement serialize(TreeNode root) which converts a tree into a string, and deserialize(String data) which converts a string back to a tree.

Constraints

The encoded string should be as compact as possible.

1. Problem Statement

Implement a trie with insert, search, and startsWith methods.

insert(word): Inserts a word into the trie.

search(word): Returns if the word is in the trie.

startsWith(prefix): Returns if there is any word in the trie that starts with the given prefix.

#### Constraints

You may assume that all inputs are consist of lowercase letters a-z.

All inputs are guaranteed to be non-empty strings.

1. Given a string containing just the characters '(', ')', '{', '}', '[', and ']', determine if the input string is valid. An input string is valid if:

Open brackets must be closed by the same type of brackets.

Open brackets must be closed in the correct order.

1. Given n non-negative integers a1, a2, ..., an , where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of the line i are at (i, ai) and (i, 0). Find two lines, which together with the x-axis forms a container, such that the container contains the most water.
2. Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.
3. Design an interval tree to efficiently find all intervals that overlap with a given interval. Implement the following operations:

insertInterval(int start, int end): Insert a new interval [start, end] into the tree.

deleteInterval(int start, int end): Delete an interval [start, end] from the tree.

findOverlappingIntervals(int start, int end): Return a list of all intervals that overlap with the interval [start, end].

#### Constraints

The intervals are represented as pairs of integers [start, end] where start ≤ end

1. Write a Java program that checks if a given string is a palindrome. A palindrome is a word, phrase, number, or other sequences of characters that reads the same forward and backward (ignoring spaces, punctuation, and capitalization).