COVER PAGE

## Programming session

* There are two programming question difficulties.
* Candidate must finish **1 easy question and 1 moderate question.**
* Candidate can implement solutions in **Java, Kotlin, Python, Python3, Swift, and JavaScript.**
* Candidate must implement solutions in the selected language **template file.**
* Candidate can use the internet.
* If the questions aren’t clear or ambiguous, feel free to ask interviewers.
* Candidate should manage your time wisely, focus on finishing both questions and do the refactoring later.
* The submitted solutions must be **runnable**.
* Focus on logic and print result in **console**. **No need** to implement user interface.

DIFFICULTY: **EASY**

**CANDIDATE CHOOSE TO FINISH ONLY ONE QUESTION**

**QUESTION EASY#1**

Implement a method *easy1* that given an integer array, which each element appears twice except for one. Find that one

|  |
| --- |
| **You must solve it with O(n) time-complexity** |

Example #1

Input: [1, 1, 2, 2, 3]

Output: 3

Example #2

Input: [-1, 2, 4, 2, -1]

Output: 4

(Optional) bonus point: solve with O(1) space-complexity

**QUESTION EASY#2**

When climbing a stair, you can either take 1 step or 2 steps.

Implement a method *easy2* that given a number of steps of a stair, returns how many distinct ways to reach the top

Example

Input: 3

Output: 3

( 1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step)

DIFFICULTY**: MODERATE**

**CANDIDATE CHOOSE TO FINISH ONLY ONE QUESTION**

**QUESTION MODERATE#1**

Implement a method *moderate1* that given an array of integers *nums*, returns an array of integers *output* where *output[i]* = product of all elements in *nums* except *nums[i]*

|  |
| --- |
| **You must solve it with O(n) time-complexity and without using division operation** |

Example

Input: [1, 2, 3, 4]

Output: [24, 12, 8, 6]

From the example, *output[1]* = product of all elements in nums except *nums[1]* which is 1 \* 3 \* 4 = 12

(Optional) bonus point: solve with O(1) space-complexity (except the *output* array)

**QUESTION MODERATE#2**

Implement a method *moderate2* that given an integer which is the number of parentheses, returns a set (array) of strings of all possible well-formed combinations

Example #1:

Input: 2

Output: [“()()”, “(())”]

Example #2:

Input: 3

Output: [“((()))”, “(()())”, “()()()”, “()(())”, “(())()”]