PROBLEM SOLVING

LeetCode

Two Sum

Problem Description:

https://leetcode.com/problems/two-sum/

Solution:

```
class Solution {
  List<int> twoSum(List<int> nums, int target) {
    // create a Map to store the complement of each number
    var numsMap = <int, int>{};

    // iterate through the list
    for (var i = 0; i < nums.length; i++) {
      // calculate the complement of the current element
      var complement = target - nums[i];

      // if the complement is already in the map, return the
indices
      if (numsMap.containsKey(complement)) {
        return [numsMap[complement]!, i];
      }

      // otherwise, add the current number and its index to the
map
      numsMap[nums[i]] = i;
    }

    // if no solution is found, return an empty list
    return [];
  }
}
```

Palindrome Number

Problem Description:

https://leetcode.com/problems/palindrome-number/

Solution:

```dart
class Solution {
  bool isPalindrome(int x) {
    return x.toString().split('').reversed.join() == x.toString();
  }
}
```

Roman to Integer

Problem Description:

https://leetcode.com/problems/roman-to-integer/description/

Solution:

```dart
class Solution {
  int romanToInt(String s) {
    Map<String, int> romanSymbols = {
      'I': 1,
      'V': 5,
      'X': 10,
      'L': 50,
      'C': 100,
      'D': 500,
      'M': 1000
    };

    int result = 0;

    for (int i = 0; i < s.length; i++) {
      if (i > 0 && romanSymbols[s[i]]! > romanSymbols[s[i - 1]]!)
{
        result += romanSymbols[s[i]]! - 2 * romanSymbols[s[i -
1]]!;
      } else {
        result += romanSymbols[s[i]]!;
      }
    }

    return result;
  }
}
```

Longest Common Prefix

Problem Description:

https://leetcode.com/problems/longest-common-prefix/description/

Solution:

```
class Solution {
  String longestCommonPrefix(List<String> strs) {
    if (strs.isEmpty)
      return '';
    else {
      String longestCommonPrefix = strs[0];

      for (int i = 1; i < strs.length; i++) {
        while (strs[i].indexOf(longestCommonPrefix) != 0) {
          // remove the last letter until indexOf == 1 (There's a
match)
          longestCommonPrefix =
              longestCommonPrefix.substring(0,
longestCommonPrefix.length - 1);

          // if no match & longestCommonPrefix become
          if (longestCommonPrefix.isEmpty) return '';
        }
      }
      return longestCommonPrefix;
    }
  }
}
```

Valid Parentheses

Problem Description:

https://leetcode.com/problems/valid-parentheses/

Solution:

```
class Solution {
  bool isValid(String string) {
```

```
    List<String> stack = <String>[];

    for (int i = 0; i < string.length; i++) {
      String bracket = string[i];

      if (bracket == '(' || bracket == '{' || bracket == '[') {
        stack.add(bracket);
      } else if (bracket == ')' && (stack.isEmpty || stack.last !=
'(')) {
        return false;
      } else if (bracket == '}' && (stack.isEmpty || stack.last !=
'{')) {
        return false;
      } else if (bracket == ']' && (stack.isEmpty || stack.last !=
'[')) {
        return false;
      } else {
        stack.removeLast();
      }
    }

    return stack.isEmpty;
  }
}
```