# SP 2023 Greedy & Divide and Conquer

# Question 1

## Problem statement

Question 1

## Class Definition

**Class:** `Question1`

**Method:** `maxPoint`

The class `Question1` includes the method `maxPoint` which accepts one `String` parameter and returns an `int`.

```
//java
public int maxPoint(String s)

//cpp
int maxPoint(string s)

#python
def maxPoint(s: str) -> int
```

## Constraints

- `s` length can be up to 20 characters.

# Examples

### Example 0

```
Question1 q1 = new Question1();
System.out.println(q1.maxPoint("XYXY")); // Returns: 2
```

XYXY ⇒ YZXY by edit the first “XY” ⇒ YZYZ, by replace the second “XY”.

### Example 1

```
Question1 q1 = new Question1();
System.out.println(q1.maxPoint("YXYXYX")); // Returns: 3
```

YXYXYX ⇒ ZYYXYX ⇒ ZYZYYX ⇒ ZYZYZY

# Test Cases

| # | Test case | Expected Result |
|---|---|---|
| 1 | XYXY | 2 |
| 2 | YXYXYX | 3 |
| 3 | XXXXXXXX | 0 |
| 4 | YYYYYYYY | |
| 5 | XYXYXYXY | |
| 6 | YXYXYXYX | |
| 7 | XXXXYXXX | |
| 8 | YYYYXYYY | |
| 9 | XYXYXYYX | |
| 10 | YXXYYXXY | |

| # | Test case | Expected Result |
|---|---|---|
| 11 | XXYXXYXX | |
| 12 | YYXYYXYY | |
| 13 | XYXYYXYXYXYX | |
| 14 | YXYXYXYXYXYX | |
| 15 | XXXXXYXXXXYXXXXX | |
| 16 | YYYYYXYYYYXYYYYY | |

Please note that the expected outputs for some test cases are not provided.

# Question 2

## Problem Statement

Question 2

## Class Definition

**Class:** `Question2`

**Method:** `eleganceTest`

The class `Question2` includes the method `eleganceTest` which accepts two `int` array parameters and returns a `boolean` array.

```
//java
public boolean[] eleganceTest(int[] nums, int[] queries)
```

```cpp
//cpp
vector<bool> eleganceTest(vector<int> nums, vector<int> queries)
```

```python
#python
def eleganceTest(nums: List[int], queries: List[int]) -> List[bool]:
```

## Constraints

- Both `nums` and `queries` arrays can have up to 20 elements on both arrays

## Examples

### Example 0

```
Question2 q2 = new Question2();
// Returns: [true, false, true]
System.out.println(q2.eleganceTest([1, 2, 3, 4, 5], [3, 8, 9]));
```

1. We can get an array of the sum 3, in the following way:
    1. nums = [1,2,3,4,5], mid = (1+5) / 2 = 3 ⇒ left = [1,2,3], right = [4,5]. We choose to keep the left array.
    2. nums = [1,2,3], mid = (1+3) / 2 = 2 ⇒ left = [1,2], right = [3]. We choose to keep the right array with the sum equalling 3.
2. It can be demonstrated that an array with the sum = 8 is impossible to generate.
3. An array with the sum = 9 can be generated in the following way:
    1. nums = [1,2,3,4,5], mid = (1+5) / 2 = 3 ⇒ left = [1,2,3], right = [4,5]. We choose to keep the right array with the sum equalling 9.

### Example 1

```
Question2 q2 = new Question2();
//Returns [false, false, true]
System.out.println(q2.eleganceTest([3, 1, 3, 1, 3], [1, 3, 11]));
```

We can get an array with the sum = 11 with zero slicing operations, because array sum is equal to 11.

# Test Cases

| Nums | Queries | Expected Result |
|---|---|---|
| {1, 2, 3, 4, 5} | {3, 8, 9} | {true, false, true} |
| {3, 1, 3, 1, 3} | {1, 3, 11} | {false, false, true} |
| {1, 1, 2, 3, 5, 8, 13} | {5, 21, 8} | {true, true, true} |
| {2, 4, 6, 8, 10, 12, 14, 16, 18, 20} | {10, 20, 30, 40} | |
| {1, 2, 3, 4, 5, 6, 7, 8} | {8, 16, 24, 32} | |
| {1, 1, 2, 2, 3, 3} | {2, 4, 6} | |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15} | {15, 30, 45, 60, 75} | |
| {1, 3, 5, 7, 9, 11, 13, 15, 17} | {17, 34, 51, 68} | |
| {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12} | {12, 24, 36, 48} | |
| {5, 10, 15, 20, 25, 30, 35} | {35, 70, 105} | |
| {2, 4, 6, 8, 10, 12, 14, 16} | {16, 32, 48} | |

Please note that the expected outputs for some test cases are not provided.