프로그래머스 -배열 회전시키기

태그

Arrays.stream(numbers).boxed().collect(Collectors.toList())

list.add(list.size() list.get(0)) list.get(list.size() - 1)

list.stream().mapToInt(Integer::intValue).toArray()

numbers.push(numbers.shift()) numbers.shift numbers.unshift

numbers.unshift(numbers.pop())

```
자바스크립트
배열 회전시키기
문제 설명
정수가 담긴 배열 numbers와 문자열 direction가 매개변수로 주어집니다. 배열 numbers의 원소를 direction 방향으로 한 칸씩 회전시킨 배열을 return하도록 solution 함수를 완성해주세요.

제한사항
3 ≤ numbers의 길이 ≤ 20
direction은 "left" 와 "right" 둘 중 하나입니다.

입출력 예 #1
numbers:[1, 2, 3] ,direction:"right"  ,result:[3, 1, 2]
numbers 가 [1, 2, 3]이고 direction이 "right" 이므로 오른쪽으로 한 칸씩 회전시킨 [3, 1, 2]를 return합니다.
입출력 예 #2
numbers:[4, 455, 6, 4, -1, 45, 6] ,direction:"left" ,result:[455, 6, 4, -1, 45, 6, 4]
numbers 가 [4, 455, 6, 4, -1, 45, 6]이고 direction이 "left" 이므로 왼쪽으로 한 칸씩 회전시킨 [455, 6, 4, -1, 45, 6, 4]를 return합니다.
```

```
function solution(numbers, direction) {
  if (direction === "right") {
    // 배열을 오른쪽으로 회전
    numbers.unshift(numbers.pop());
  } else {
    // 배열을 왼쪽으로 회전
    numbers.push(numbers.shift());
  }
  return numbers;
}
```

```
 이 문제를 해결하기 위해 JavaScript Array 객체의 shift() 및 unshift() 메서드를 사용하여 배열의 요소를 왼쪽 또는 오른쪽으로 회전할 수 있습니다. 이 솔루션은 방향 값을 확인하기 위해 조건문을 사용합니다. 방향이 "right"이면 unshift() 메서드를 사용하여 배열의 마지막 요소를 배열 앞에 추가하고 pop() 메서드를 사용하여 배열에서 마지막 요소를 제거합니다. 방향이 "왼쪽"인 경우 shift() 메서드를 사용하여 배열의 첫 번째 요소를 제거하고 push() 메서드를 사용하여 배열 끝에 추가합니다.
```

numbers.push(numbers.shift()) 는 숫자 배열의 요소를 왼쪽으로 한 단위 회전시키는 코드입니다. shift() 메서드는 숫자 배열의 첫 번째 요소를 제거하고 반환합니다. push() 메서드는 요소를 배열 끝에 추가합니다. 예를 들어 숫자가 [1, 2, 3]인 경우 numbers.shift()는 배열에서 첫 번째 요소 1을 제거하여 반환하고 numbers.push(1)는 배열 끝에 요소 1을 추가하여 결과를 얻습니다. 숫자는 [2, 3, 1]입니다.

numbers.unshift(numbers.pop()) 는 숫자 배열의 요소를 오른쪽으로 한 단위 회전시키는 코드입니다.pop() 메서드는 숫자 배열의 마지막 요소를 제거하고 반환합니다. unshift() 메서드는 요소를 배열 앞에 추가합니다.예를 들어 숫자가 [1, 2, 3]인 경우 numbers.pop()은 배열에서 마지막 요소 3을 제거하고 반환하고 numbers.unshift(3)는 요소 3을 배열 앞에 추가합니다. 숫자는 [3, 1, 2]입니다.

```
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

class Solution {
    public int[] solution(int[] numbers, String direction) {
        List<Integer> list = Arrays.stream(numbers).boxed().collect(Collectors.toList
());

        if (direction.equals("right")) {
            list.add(0, list.get(list.size() - 1)); // 목록의 마지막 요소를 목록의 시작 부분에
추가한 다음 마지막 요소를 제거
            list.remove(list.size() - 1);
        } else {
            list.add(list.size(), list.get(0));
            list.remove(0);
        }
        return list.stream().mapToInt(Integer::intValue).toArray();
    }
}
```

정수 숫자 배열과 문자열 방향을 매개 변수로 사용하고 숫자 요소가 방향으로 지정된 방향으로 한 단위 회전된 배열을 반환하는 솔루션() 메서드를 사용하여 Solution 클래스를 정의합니다. Arrays.stream() 및 Collectors.toList() 메서드를 사용하여 배열 번호를 List 객체로 변환합니다. 그런 다음 조건문을 사용하여 방향 값을 확인합니다. 방향이 "right"이면 add() 및 remove() 메서드를 사용하여 목록의 요소를 오른쪽으로 한 단위 회전합니다. 방향이 "왼쪽"인 경우 동일한 메서드를 사용하여 목록의 요소를 왼쪽으로 한 단위 회전합니다. 마지막으로 이 메서드는 stream(), mapToInt() 및 toArray() 메서드를 사용하여 목록을 다시 배열로 변환하고 결과를 반환합니다.

Arrays.stream() 메서드를 사용하여 1)숫자 배열에서 스트림을 만든 다음 2)boxed() 메서드를 사용하여 스트림의 요소를 프리미티브에서 해당 래퍼 유형(이 경우 Integer)으로 변환합니다. 3)마지막으로, collect() 메서드는 스트림의 요소를 List<Integer>로 수집하는 데 사용됩니다 결과 목록에는 동일한 순서로 숫자 배열의 요소가 포함됩니다.
Arrays.stream(numbers)은 numbers 배열에 있는 요소의 스트림을 만듭니다. .boxed()는 스트림의 요소를 프리미티브에서 해당 래퍼 유형으로 변환합니다. 예를 들어 int 값은 Integer 개체로 변환됩니다.
.collect(Collectors.toList())는 스트림의 요소를 목록으로 수집합니다. Collectors.toList() 메서드는 입력 요소를 새 목록에 누적하는 수집기를 반환합니다. 따라서 이 코드는 기본적으로 각 요소가 해당 래퍼 유형으로 변환된 숫자 배열의 요소 목록을 생성합니다.

`list.get(list.size() - 1)` 목록의 마지막 요소를 가져옵니다. `list.add(0, list.get(list.size() - 1))` 목록의 시작 부분에 마지막 요소를 추가합니다. `add()` 메서드는 요소를 추가할 인덱스와 추가할 요소의 두 가지 인수를 사용합니다. 이 경우 요소는 인덱스 `0`(목록의 시작 부분)에 추가됩니다.

`list.remove(list.size() - 1)` 목록의 마지막 요소를 제거합니다. `remove()` 메서드는 인덱스를 인수로 사용하고 해당 인덱스에서 요소를 제거합니다. 이 경우 인덱스는 목록의 마지막 요소입니다(`list.size() - 1`). 따라서 이 코드는 기본적으로 목록의 마지막 요소를 목록의 시작 부분으로 이동합니다. 결과 목록은 원래 목록의 시작 부분에 있던 요소가 이제 끝에 있는 요소와 함께 오른쪽으로 한 위치 이동된 모든 요소를 갖게 됩니다.