Codility_

CodeCheck Report: trainingVJ2JM6-YRG

Test Name:

Check out C

Summary Timeline

Tasks summary

Task	Time spent	Score
CyclicRotation Go	43 min	50%

Total score

Tasks Details

1. CyclicRotation

Easy

Rotate an array to the right by a given number of steps.

Task Score	Correctness	Performance
50%	50%	Not assessed

Task description

An array A consisting of N integers is given. Rotation of the array means that each element is shifted right by one index, and the last element of the array is moved to the first place. For example, the rotation of array A = [3, 8, 9, 7, 6] is [6, 3, 8, 9, 7] (elements are shifted right by one index and 6 is moved to the first place).

The goal is to rotate array A K times; that is, each element of A will be shifted to the right K times.

Write a function:

```
func Solution(A []int, K int) []int
```

Solution

Programming language used:	Go
Total time used:	43 minutes
Effective time used:	43 minutes
Notes:	*not defined yet*

made:

```
[3, 8, 9, 7, 6] -> [6, 3, 8, 9, 7]
[6, 3, 8, 9, 7] -> [7, 6, 3, 8, 9]
[7, 6, 3, 8, 9] -> [9, 7, 6, 3, 8]
```

For another example, given

```
A = [0, 0, 0]
K = 1
```

the function should return [0, 0, 0]

Given

```
A = [1, 2, 3, 4]
K = 4
```

the function should return [1, 2, 3, 4]

Assume that:

- N and K are integers within the range [0..100];
- each element of array A is an integer within the range [−1,000..1,000].

In your solution, focus on **correctness**. The performance of your solution will not be the focus of the assessment.

final, score: **50**

```go
package solution

import "math"

func rotateRight(A []int, K int) []i
    lastKElementsToPutInFront := A[l
    return append(lastKElementsToPut
}

func rotateLeft(A []int, K int) []in
    firstKElementsToPutInBack := A[:
    return append(A[K:], firstKEleme
}

func Solution(A []int, K int) []int
    if len(A) == K {
        return A
    }

    KLeft := len(A) - K

    minimumK := math.Min(float64(K),

    if minimumK == float64(K) {
        return rotateRight(A, K)
    }

    return rotateLeft(A, KLeft)
}
```

Analysis summary

The following issues have been detected: runtime

For example, for the input `([], 1)` the solution t
unexpectedly.

Analysis

collapse all	Example tests	
	example first example test	✔ OK
1.	0.001 s **OK**	
	example2 second example test	✔ OK
1.	0.001 s **OK**	
	example3 third example test	✔ OK

2. 0.001 s **RUNTIME ERROR**, tested program termina
   code 2

   stderr:

```
panic: runtime error: slice bounds ou

goroutine 1 [running]:
solution.Solution(0x573a20, 0x0, 0x0,
```

	single one element, 0 <= K <= 5	✘ **RUNTIM** tested proc terminated 2

1. 0.001 s **OK**
2. 0.001 s **OK**
3. 0.001 s **RUNTIME ERROR**, tested program termina
   code 2

   stderr:

```
panic: runtime error: slice bounds ou

goroutine 1 [running]:
solution.Solution(0xc208032288, 0x1,
```

	double two elements, K <= N	✔ **OK**

1. 0.001 s **OK**
2. 0.001 s **OK**

	small1 small functional tests, K < N	✔ **OK**

1. 0.001 s **OK**
2. 0.001 s **OK**

	small2 small functional tests, K >= N	✘ **RUNTIM** tested proc terminated 2

1. 0.001 s **OK**
2. 0.001 s **RUNTIME ERROR**, tested program termina
   code 2

   stderr:

```
panic: runtime error: slice bounds ou

goroutine 1 [running]:
solution.Solution(0xc20800a2d0, 0x6,
```

3. 0.001 s **RUNTIME ERROR**, tested program termina
   code 2

   stderr:
   ```
   panic: runtime error: slice bounds ou

   goroutine 1 [running]:
   solution.Solution(0xc20800a2d0, 0x5, (
   ```

small_random_all_rotations ✔ **OK**
small random sequence, all rotations, N = 15

1. 0.001 s **OK**
2. 0.001 s **OK**
3. 0.001 s **OK**
4. 0.001 s **OK**
5. 0.001 s **OK**
6. 0.001 s **OK**
7. 0.001 s **OK**
8. 0.001 s **OK**
9. 0.001 s **OK**
10. 0.001 s **OK**
11. 0.001 s **OK**
12. 0.001 s **OK**
13. 0.001 s **OK**
14. 0.001 s **OK**
15. 0.001 s **OK**

medium_random ✔ **OK**
medium random sequence, N = 100

1. 0.001 s **OK**
2. 0.001 s **OK**

maximal ✘ **RUNTIM**
maximal N and K
tested proc
terminated
2

1. 0.001 s **OK**
2. 0.001 s **RUNTIME ERROR**, tested program termina
   code 2

.

3. 0.001 s **OK**
4. 0.001 s **OK**