Tutorial for Problem B - Array Rotations

Naive Solution

The simplest solution is to rotate the array once at a time. For this, we can use a temporary array. For each rotation, we copy the last element of the array to the first position of the temporary array, and then copy all other elements of the array to the remaining positions. Finally, we copy all elements of the temporary array to the original array.

The important observation is that we only need to rotate the array $k \mod n$ times, since rotating the array n times will result in the same array.

This solution is $O(n \cdot k)$.

Code

```
#include <stdio.h>

#define MAX_SIZE 1'000

int a[MAX_SIZE + 10];
int temp[MAX_SIZE + 10];

int main() {

    int n, k;
    scanf("%d %d", &n, &k);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }

    k = k % n;

    // rotate k times
    while (k--) {
        // copy last element to temp[0]
        temp[0] = a[n - 1];
        // copy all elements from a[0 .. n - 2] to temp[1 .. n - 1]
        for (int i = 0; i < n - 1; ++i) {
            temp[i + 1] = a[i];
        }
        // copy all elements from temp back to a
        for (int i = 0; i < n; ++i) {
            a[i] = temp[i];
        }
    }
```

```
    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }

    printf("\n");

    return 0;
}
```

Better Solution

We can do better than the naive solution. Instead of rotating the array k times, we can rotate the array only once. For this, we can use a temporary array. We copy the first k elements of the array to the temporary array, and then copy the remaining elements of the array to the first $n - k$ positions. Finally, we copy all elements of the temporary array to the remaining positions.

This solution is $O(n)$.

Code

```
#include <stdio.h>

#define MAX_SIZE 1'000

int a[MAX_SIZE + 10];
int temp[MAX_SIZE + 10];

int main() {

    int n, k;
    scanf("%d %d", &n, &k);
    for (int i = 0; i < n; ++i) {
        scanf("%d", &a[i]);
    }

    k = k % n;

    // copy k elements from the end of a to the beginning of temp
    for (int i = 0; i < k; ++i) {
        temp[i] = a[n - k + i];
    }
    // copy the remaining (n - k) elements to the end of temp
    for (int i = k; i < n; ++i) {
        temp[i] = a[i - k];
    }
```

```
    // copy all elements from temp back to a
    for (int i = 0; i < n; ++i) {
        a[i] = temp[i];
    }


    for (int i = 0; i < n; ++i) {
        printf("%d ", a[i]);
    }

    printf("\n");

    return 0;
}
```