Day 4: Rotate an Array

Gati Goyal

"A good programmer looks for patterns and simplifies the code."
— Anonymous

---

1 Introduction

Array rotation is a frequently encountered problem in programming where the elements of an array are shifted by a specified number of positions. This document focuses on rotating an array to the left by k positions using an efficient three-step reversal method.

2 Problem Statement

Problem: Rotate an array to the left by k positions. **Hint:** Break the problem into three steps using reversal:

1. Reverse the first k elements.
2. Reverse the remaining elements.
3. Reverse the entire array.

3 Algorithm

3.1 Three-Step Reversal Method

1. Reverse the first k elements:
   - Swap the first and kth element, then move inward.
2. Reverse the remaining elements:
   - Swap the elements from the $k+1$th index to the end.
3. Reverse the entire array:
   - Swap the first and last elements of the full array and move inward.

4 Code

```
#include <stdio.h>

// Function to reverse a portion of the array
void reverse(int arr[], int start, int end) {
    while (start < end) {
        int temp = arr[start];
        arr[start] = arr[end];
        arr[end] = temp;
        start++;
        end--;
    }
}

// Function to rotate the array to the left by k positions
void rotateArray(int arr[], int n, int k) {
    // Normalize k to prevent unnecessary rotations
    k = k % n;

    // Step 1: Reverse the first k elements
    reverse(arr, 0, k - 1);

    // Step 2: Reverse the remaining elements
    reverse(arr, k, n - 1);

    // Step 3: Reverse the entire array
    reverse(arr, 0, n - 1);
}

int main() {
    int n, k;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the value of k: ");
    scanf("%d", &k);

    rotateArray(arr, n, k);

    printf("Rotated Array:\n");
    for (int i = 0; i < n; i++) {
```

```
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

5 Step-by-Step Explanation

1. Reverse the first k elements:
   - For $k = 3$ and array $[1, 2, 3, 4, 5]$, reverse $[1, 2, 3]$ to $[3, 2, 1]$.
2. Reverse the remaining elements:
   - Reverse $[4, 5]$ to $[5, 4]$.
3. Reverse the entire array:
   - Reverse $[3, 2, 1, 5, 4]$ to $[4, 5, 1, 2, 3]$.

6 Complexity Analysis

6.1 Time Complexity

- Each reversal involves $O(n)$ swaps.
- Total time complexity is $O(n)$.

6.2 Space Complexity

- In-place rotation ensures space complexity of $O(1)$.

7 Rotation vs Reversal

Criteria	Rotation	Reversal
Definition	Shifts elements by k positions	Flips elements in a range
Memory Usage	Depends on method	In-place
Applications	Circular queues, buffers	Sorting, array rotation

8 Conclusion

The three-step reversal method provides an efficient way to rotate an array to the left by k positions in-place, minimizing both time and space complexity. This technique is widely applicable in various real-world problems like buffer management and circular queues.

9 Output

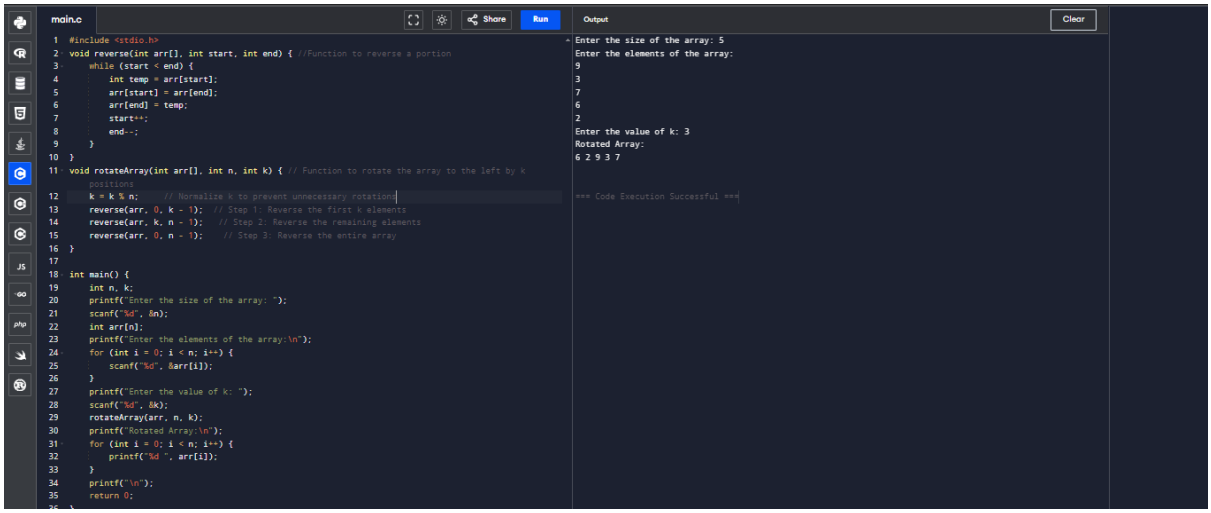

Figure 1: Output in online compiler