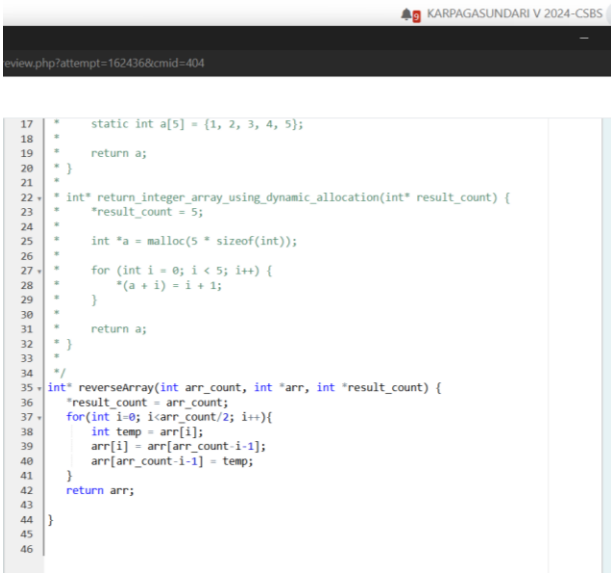
WEEK 15

1. Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

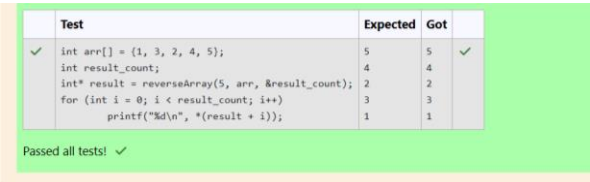
Input Format For Custom Testing : The first line contains an integer, n, the number of elements in arr. Each line i of the n subsequent lines (where 0 ≤ i < n) contains an integer, arr[i].

Program:




```
 *     static int a[5] = {1, 2, 3, 4, 5};
 *
 *     return a;
 * }
 *
 * int* return_integer_array_using_dynamic_allocation(int* result_count) {
 *     *result_count = 5;
 *
 *     int *a = malloc(5 * sizeof(int));
 *
 *     for (int i = 0; i < 5; i++) {
 *         *(a + i) = i + 1;
 *     }
 *
 *     return a;
 * }
 *
 */
int* reverseArray(int arr_count, int *arr, int *result_count) {
   *result_count = arr_count;
   for(int i=0; i<arr_count/2; i++){
       int temp = arr[i];
       arr[i] = arr[arr_count-i-1];
       arr[arr_count-i-1] = temp;
   }
   return arr;
}
```

Output:



	Test	Expected	Got	

✓	int arr[] = {1, 3, 2, 4, 5};
	int result_count;
	int* result = reverseArray(5, arr, &result_count);
	for (int i = 0; i < result_count; i++)
	printf("%d\n", *(result + i));
		5
		4
		2
		3
		1
			5
			4
			2
			3
			1
				✓

Passed all tests! ✓

2. An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of minLength or more, and it can only make one cut at a time. Given the array lengths representing the desired lengths of each segment, determine if it is possible to make the necessary cuts using this machine. The rod is marked into lengths already, in the order given. . Function Description Complete the function cutThemAll in the editor below. cutThemAll has the following parameter(s): int lengths[n]: the lengths of the segments, in order int minLength: the minimum length the machine can accept

Program:


```
/*
 * To return the string from the function, you should either do static allocation or dyna
 *
 * For example,
 * char* return_string_using_static_allocation() {
 *     static char s[] = "static allocation of string";
 *
 *     return s;
 * }
 *
 * char* return_string_using_dynamic_allocation() {
 *     char* s = malloc(100 * sizeof(char));
 *
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
 */
char* cutThemAll(int lengths_count, long *lengths, long minLength) {
    long t=0, i=1;
    for(int i=0; i<lengths_count; i++){
        t+= lengths[i];
    }
    do{
        if(t - lengths[lengths_count-i-1]<minLength){
            return "Impossible";
        }
        i++;
    }while(i<lengths_count-1);
    return "Possible";
    }
```

Output:

	Test	Expected	Got	
✓	long lengths[] = {3, 5, 4, 3}; printf("%s", cutThemAll(4, lengths, 9))	Possible	Possible	✓
✓	long lengths[] = {5, 6, 2}; printf("%s", cutThemAll(3, lengths, 12))	Impossible	Impossible	✓

Passed all tests! ✓