Q1)Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

## Example

$$arr = [1, 3, 2, 4, 5]$$

Return the array [5, 4, 2, 3, 1] which is the reverse of the input array.

**Function Description** 

Complete the function reverseArray in the editor below.

reverseArray has the following parameter(s):

int arr[n]: an array of integers

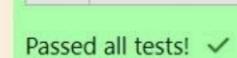
Return

int[n]: the array in reverse order

Constraints

```
int* reverseArray(int arr count, int *arr, int *result count) {
    int *b = (int *)malloc(arr count * sizeof(int));
   if(b == NULL){
       return NULL;
    *result count = arr count;
    for(int i =0;i<arr count;i++){
       b[i]=arr[arr count -1 -i];
    return b;
```

	Test	Expected	Got	
~	int arr[] = {1, 3, 2, 4, 5};	5	5	~
	int result_count;	4	4	
	<pre>int* result = reverseArray(5, arr, &amp;result_count);</pre>	2	2	
	for (int i = 0; i < result_count; i++)	3	3	
	printf("%d\n", *(result + i));	1	1	



```
char* cutThemAll(int lengths_count, long *lengths, long minLength) {
    long totallength=0;
    for (int i=0;i< lengths_count;i++){
        totallength += lengths[i];
    if(totallength<minLength){</pre>
        return "Impossible";
    long remainingLength=totallength;
    for(int i=0;i<lengths_count;i++){
        remainingLength-=lengths[i];
        if(remainingLength>=minLength){
            return "Possible";
    return "Impossible";
```

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

Passed all tests! ✓