```
 *     }
 *
 *     return a;
 * }
 *
 */
#include<stdio.h>
#include<stdlib.h>
int* reverseArray(int arr_count, int *arr, int *result_count) {
   *result_count=arr_count;
    int *reversed=(int *)malloc(arr_count * sizeof(int));
    if(reversed==NULL)
    {
       exit(1);
    }
    for(int i=0;i<arr_count;i++)
    {
        reversed[i]=arr[arr_count-1-i];
    }
     return reversed;
}
```

	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! ✓

```
 *     s = "dynamic allocation of string";
 *
 *     return s;
 * }
 *
 */
#include<stdio.h>
char* cutThemAll(int lengths_count, long *lengths, long minLength) {
   long totalLength=0;

   for(int i=0;i<lengths_count;i++)
   {
       totalLength+=lengths[i];
   }
   long currentLength=0;
   for(int i=0;i<lengths_count -1;i++)
   {
       currentLength+=lengths[i];
       long remainingLength=totalLength-currentLength;
       if(remainingLength > minLength)
       {
           return "Possible";
       }
   }
    return "Impossible";
}
```

	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! ✓