**Question 1**

Correct

Marked out of 1.00

⚑ Flag question

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

$1 \le n \le 100$

$0 < arr[i] \le 100$

### 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 $0 \le i < n$) contains an integer, *arr[i]*.

### Sample Case 0

### Sample Input For Custom Testing

5

1

3

2

4

5

### Sample Output

5

4

2

3

1

### Explanation

The input array is [1, 3, 2, 4, 5], so the reverse of the input array is [5, 4, 2, 3, 1].

### Sample Case 1

4

17

10

21

45

Sample Output

45

21

10

17

Explanation

The input array is [17, 10, 21, 45], so the reverse of the
input array is [45, 21, 10, 17].

**Answer:** (penalty regime: 0 %)

Reset answer

```
 1
 2    Complete the 'reverseArray' function belo
 3
 4    The function is expected to return an INT
 5    The function accepts INTEGER_ARRAY arr as
 6  /
 7
 8
 9    To return the integer array from the func
10       - Store the size of the array to be r
11       - Allocate the array statically or dy
12
13    For example,
14  int* return_integer_array_using_static_al
15        *result_count = 5;
16
17        static int a[5] = {1, 2, 3, 4, 5};
18
19        return a;
20  }
21
22  int* return_integer_array_using_dynamic_a
23        *result_count = 5;
24
25        int *a = malloc(5 * sizeof(int));
26
27        for (int i = 0; i < 5; i++) {
28            *(a + i) = i + 1;
29        }
30
31        return a;
32  }
33
34  /
35  t* reverseArray(int arr_count, int *arr, i
36       *result_count = arr_count;
37       static int rev[100];
38       int i,j=0;
39       for(i=arr_count-1;i>=0;i--)
40       rev[j++]=arr[i];
41       return rev;
42
```

Question **2**

Correct

Marked out of
1.00

⚑ Flag
question

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.

**Example**

$n = 3$

lengths = [4, 3, 2]

minLength = 7

The rod is initially sum(lengths) = 4 + 3 + 2 = 9 units long. First cut off the segment of length 4 + 3 = 7 leaving a rod 9 - 7 = 2. Then check that the length 7 rod can be cut into segments of lengths 4 and 3. Since 7 is greater than or equal to minLength = 7, the final cut can be made. Return "Possible".

**Example**

$n = 3$

lengths = [4, 2, 3]

minLength = 7

The rod is initially sum(lengths) = 4 + 2 + 3 = 9 units long. In this case, the initial cut can be of length 4 or 4 + 2 = 6. Regardless of the length of the first cut, the remaining piece will be shorter than minLength. Because $n - 1 = 2$ cuts cannot be made, the answer is "Impossible".

**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

Returns

string: *"Possible"* if all *n-1* cuts can be made. Otherwise, return the string *"Impossible"*.

Constraints

- $2 \le n \le 10^5$
- $1 \le t \le 10^9$
- $1 \le lengths[i] \le 10^9$
- The sum of the elements of lengths equals the uncut rod length.

**Input Format For Custom Testing**

The first line contains an integer, *n*, the number of elements in *lengths*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains an integer, *lengths[i]*.

The next line contains an integer, *minLength*, the minimum length accepted by the machine.

**Sample Case 0**

**Sample Input For Custom Testing**

| STDIN | Function |
| ----- | -------- |
| 4 | → lengths[] size n = 4 |
| 3 | → lengths[] = [3, 5, 4, 3] |
| 5 | |
| 4 | |
| 3 | |
| 9 | → minLength= 9 |

## Sample Output

Possible

## Explanation

The uncut rod is 3 + 5 + 4 + 3 = 15 units long. Cut the rod into lengths of 3 + 5 + 4 = 12 and 3. Then cut the 12 unit piece into lengths 3 and 5 + 4 = 9. The remaining segment is 5 + 4 = 9 units and that is long enough to make the final cut.

## Sample Case 1
### Sample Input For Custom Testing

```
STDIN   Function
-----   --------
3    →  lengths[] size n = 3
5    →  lengths[] = [5, 6, 2]
6
2
12   →  minLength= 12
```

## Sample Output

Impossible

## Explanation

The uncut rod is 5 + 6 + 2 = 13 units long. After making either cut, the rod will be too short to make the second cut.

**Answer:** (penalty regime: 0 %)

```
1  /*
2   * Complete the 'cutThemAll' function bel
3   *
4   * The function is expected to return a S
5   * The function accepts following paramet
6   *  1. LONG_INTEGER_ARRAY lengths
7   *  2. LONG_INTEGER minLength
8   */
9
10  /*
11   * To return the string from the function
12   *
```

**Answer:** (penalty regime: 0 %)

Reset answer

```
1  /*
2   * Complete the 'cutThemAll' function bel
3   *
4   * The function is expected to return a S
5   * The function accepts following paramet
6   *    1. LONG_INTEGER_ARRAY lengths
7   *    2. LONG_INTEGER minLength
8   */
9
10 /*
11  * To return the string from the function
12  *
13  * For example,
14  * char* return_string_using_static_alloc
15  *      static char s[] = "static allocati
16  *
17  *      return s;
18  * }
19  *
20  * char* return_string_using_dynamic_allo
21  *      char* s = malloc(100 * sizeof(char
22  *
23  *      s = "dynamic allocation of string"
24  *
25  *      return s;
26  * }
27  *
28  */
29 char* cutThemAll(int lengths_count, long
30     int s=0;
31     for(int i=0;i<lengths_count-1;i++)
32     {
33         s+= *(lengths +i);
34     }
35     if(s >= minLength){
36         return "Possible";
37     }
38     else{
39         return "Impossible";
40     }
41 }
42
```

| Test | E⟩ |
|---|---|
| ✓ | `long lengths[] = {3, 5, 4, 3};`<br>`printf("%s", cutThemAll(4, lengths, 9))` | Po |
| ✓ | `long lengths[] = {5, 6, 2};`<br>`printf("%s", cutThemAll(3, lengths, 12))` | Im |

Passed all tests! ✓

Finish review