

Status	Finished
Started	Wednesday, 31 December 2025, 10:57 PM
Completed	Wednesday, 31 December 2025, 11:18 PM
Duration	21 mins 13 secs

Question 1

Correct

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 $\text{sum}(\text{lengths}) = 4 + 3 + 2 = 9$ 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 $\text{sum}(\text{lengths}) = 4 + 2 + 3 = 9$ 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

- $2 \leq n \leq 10^5$
- $1 \leq t \leq 10^9$
- $1 \leq lengths[i] \leq 10^9$
- 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 $0 \leq i < n$) 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 %)

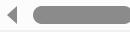
reset answer

```
1 B- /* 
2 * Complete the 'cutThemAll'
3 *
4 * The function is expected to
5 * The function accepts follow
6 * 1. LONG_INTEGER_ARRAY len
7 * 2. LONG_INTEGER minLength
8 */
9
10 B- /*
11 * To return the string from
12 *
13 * For example,
14 B- * char* return_string_using_
15 *     static char s[] = "sta
16 *
17 *     return s;
18 */
19 *
20 B- * char* return_string_using_
21 *     char* s = malloc(100 *
22 *
23 *     s = "dynamic allocatio
24 *
25 *     return s;
26 */
27 *
28 */
29 #include<stdio.h>
30 #include<stdio.h>
31 B- char* cutThemAll(int lengths_
32         long total=0;
```

```

33     for(int i=0;i<lengths_cou
34         total+=lengths[i];
35     }
36     for(int i=0;i<lengths_cou
37         long remainder=total-
38         if(remainder>=minLeng
39             char*result=(char*
40             sprintf(result,"P
41             return result;
42         }
43     }
44     char*result=(char*)malloc(
45     sprintf(result,"Impossibl
46     return result;
47 }
48
49
50

```



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

Passed all tests!

Question 2

Correct

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

$1 \leq n \leq 100$

$0 < arr[i] \leq 100$

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 $0 \leq i < n$) 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

Sample Input For Custom Testing

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 B- /*  
2 * Complete the 'reverseArray'  
3 *  
4 * The function is expected to  
5 * The function accepts INTEGER  
6 */  
7
```

```

8  /*
9   * To return the integer arra
10  *      - Store the size of th
11  *      - Allocate the array s
12  *
13  * For example,
14  * int* return_integer_array_
15  *     *result_count = 5;
16  *
17  *     static int a[5] = {1,
18  *
19  *         return a;
20  *     }
21  *
22  * int* return_integer_array_
23  *     *result_count = 5;
24  *
25  *     int *a = malloc(5 * si
26  *
27  *     for (int i = 0; i < 5;
28  *         *(a + i) = i + 1;
29  *     }
30  *
31  *     return a;
32  * }
33  *
34  */
35 #include<stdio.h>
36 int* reverseArray(int arr_cou
37     *result_count=arr_count;
38     int*reversed_arr=(int*)ma
39     for(int i=0;i<arr_count;i
40         reversed_arr[i]=arr[a
41     }
42     return reversed_arr;
43 }
44

```

	Test	Expected	Got	
nl̄	<pre> 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)); </pre>	5 4 2 3 1	5 4 2 3 1	nl̄ //

Passed all tests!

III