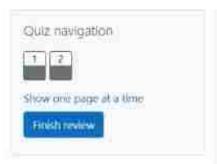
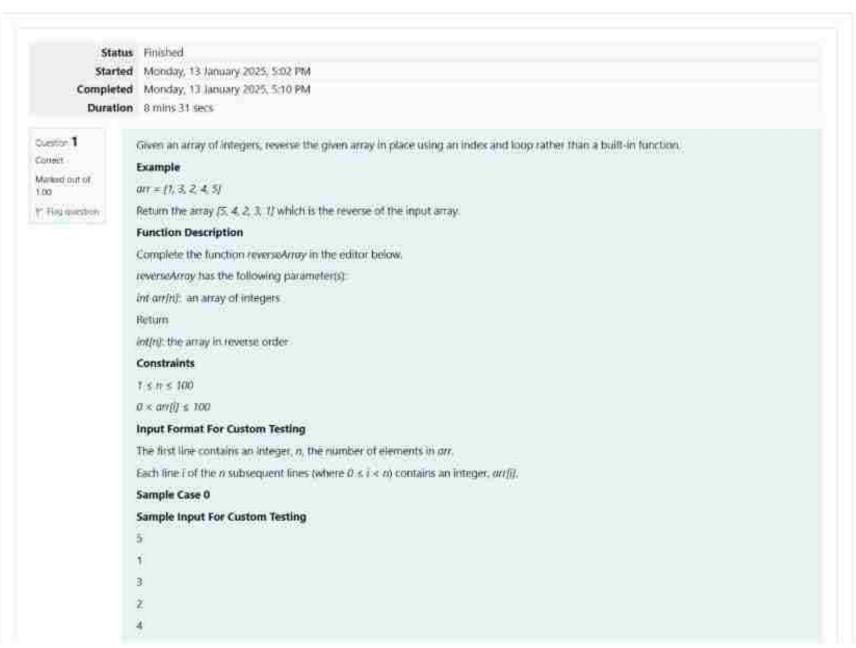
GE23131-Programming Using C-2024





```
Sample Output
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
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: 0.96)
 Reset answer.
       " complete the 'reversetriey' function below.
   3:
       " the function is expected to return an INTEGER ARMY.
       * The function accepts INTEGER ASSAY are as parameter.
   6
   2
   B3 /2
       * to return the integer array from the function, you should:
   9
   10
             - Store the size of the array to be returned in the result count variable
           - Allocate the array statically or dynamically
   11.
```

UN 100 OR 19 E BILLIE 19 O

. For example,

A SHARE THE PARTY OF THE PARTY

\$35

```
UNIT TOTAL DIRECTOR ACTOR STREET, METERS AND ACTOR ACT
15
                                               "remailt count = 5;
16
                                              wratic int a[5] = [1, 2, 8, 4, 5];
3.7
IB
19:
                                              return #1:
                      4 5
28
21
                     * int* return_integer_array_using_dynamic_allocation(int* result_count) {
22 -
23
                                              *result_count = 5;
24.
25
                                              int "a - malloc(5 * sireof(int));
26
27
                                             for (int 1 - 0) 1 + 5; i+1-1
28
                                                              *(a+1) - 1 + 3;
29
30
33
                                              metium as
                     0 3
12
33
34
35 - int" reverseArray(int arr count, int "arr, int "result_count) [
                                  "result_count_arr_count;
365
37.
                                 for(int 1 0;1 arr count/2;1++)(
                                                   int temp:arr[i];
38
115
                                                   arr[1] arr[arr_count 1-1];
40
                                                  arr[arr count 1-1] temp;
43
42
                                  return arm;
43
44
```

1	Test	Expected	Got	
4	<pre>let ure() = (1, 1, 1, 1, 4, 5);</pre>	3	8	V
	int result_count;	5	6.	
	intf result - resurceArray(5, arr, Arecult_count);	2	2	
	for (int 1 = 0; i = result_count; i++)	9	3	
	printf("Ma'n", f(result + i));	i i	1	

Conect: Marked out of 1,00

An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of mintength or more, and it can only make one cut at a time. Given the array lengths if 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

```
tt = 3

lengths = [4, 3, 2]

mintength = 7
```

The rod is initially sum(lengthi) = 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 min(length) = 7, the final cut can be made. Return "Possible".

Example

```
n = 3
lengths = [4, 2, 3]
mintength = 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 mintength. 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 mintength; 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 sns 105

- 151510
- 1 ≤ kengths/i/ ≤ 10¹¹
- The sum of the elements of lengths equals the uncut rad 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, $kingths \beta i$.

The next line contains an integer, mintength, 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]

4

3

9 → mint.ength= 9
```

Sample Output

Fossible:

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

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: 0.56)

Reset answer

```
" Complete the 'cutthemall' function below.
3 .
   7 The function is expected to return a STRING;
5 " The function accepts following parameters:
    " 1, com Bitteren Annay Dougens
    * 2. LONG INTEGER minimigth:
B
9
10 -
    * to return the string from the function, you should either do static allocation or dynamic allocation
1.1
32
    * For example,
131
14 - A chart return string using static allocation() (
          static char of ] - "static allocation of string";
15
16
17
          ration at
```

```
" char" ceters_string_using_dynamic_allocation() (
          charf a = malloc(1000 * streef(char));
23
72
23
          a - "dynamic Wincation of crolog";
24
          return st
26
27
28
    char* cutThemAll(int lengths_count, long "lengths, long minLength) {
20
        long t-0,1-1;
30
        for(int i 0;i=-lengths_count-1;i++)
33
12
            t==lengths[i];
13
54
85
        do
16
37
            if(t lengths[lengths_count 1-1] mintength)
38
                return "Impossible";
30
40
43
            1++1
42
        ]while(i<lengths_count-1);
43
        return "Possible":
44
45
46
```

	Test	Expected	Got	
4	<pre>lang langths[] = (2, 5, 4, 3); printf("No", cotThusAll(4, langths, 3))</pre>	Perallile	Pessible	~
4	long lengths[] = {5, 6, 2}; printf("%", cutThemAll(s, Lengths, 12))	Impossible	Impossible	v

Passed all testel -/