MOKSHA.K ECE-D 240801205

Question 1: Reverse a List **Problem Statement:** Given an array of integers, reverse the given array in place using an index and loop rather than a builtin 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 Input For Custom Testing 5 1 3 2

4

5

Sample Output

54231

Explanation

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

```
35 v int* reverseArray(int arr_count, int *arr, int *result_count) {
        *result_count = arr_count;
36
        for(int i = 0; i < arr_count/2; i++){</pre>
37 *
             int temp =arr[i];
38
             arr[i] = arr[arr_count-i-1];
39
             arr[arr_count-i-1] = temp;
40
41
42
        return arr;
43
44
45
```

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

Question 2:

Maximize the Value

Rearrange an array of integers so that the calculated value U is maximized. Among the

arrangements that satisfy that test, choose the array with minimal ordering. The value of U for an array with n elements is calculated as:

 $U = arr[1] \times arr[2] \times (1 \div arr[3]) \times arr[4] \times ... \times arr[n-1] \times (1 \div arr[n]) \text{ if n is odd (or)}$

 $U = arr[1] \times arr[2] \times (1 \div arr[3]) \times arr[4] \times ... \times (1 \div arr[n-1]) \times arr[n]$ if n is even

The sequence of operations is the same in either case, but the length of the array, n, determines whether the calculation ends on arr[n] or (1÷arr[n]). Arrange the elements to maximize U and the items are in the numerically smallest possible

order.

Example: arr = [5, 7, 9, 21, 34]

To maximize U and minimize the order, arrange the array as [9, 21, 5, 34, 7] so $U = 9 \times 21 \times (1 \div 5) \times 34 \times (1 \div 7) = 183.6$. The same U can be achieved using several other orders, e.g. $[21, 9, 7, 34, 5] = 21 \times 9 \times (1 \div 7) \times 34 \times (1 \div 5) = 183.6$, but they are not in the minimal order.

Function Description: Complete the function rearrange in the editor below. rearrange has the

following parameter(s): int arr[n]: an array of integers

Returns: int[n]: the elements of arr rearranged as described

Constraints: $1 \le n \le 105$, $1 \le n \le 109$

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 $1 \le i \le n$) contains an integer, arr[i].

Sample Input For Custom Testing

STDIN Function

 $4 \rightarrow arr[]$ size n = 4

 $1 \rightarrow arr = [1, 2, 3, 4]$

2

3

4

Sample Output

2

3

3

1

4

Explanation

 $U = 2 \times 3 \times (1 \div 1) \times 4 = 24$. All other arrangements where U = 24 are numerically higher than this array, e.g. [2, 3, 1, 4] < [3, 4, 1, 2].

```
29 - char* cutThemAll(int lengths_count, long *lengths, long minLength) {
30
        long t=0, i =1;
        for(int i=0; i<=lengths_count-1; i++){</pre>
31 +
            t += lengths[i];
32
33
        do{
34 +
            if(t-lengths[lengths_count-i-1] < minLength){</pre>
35 +
                 return "Impossible";
36
37
            i++;
38
39
        }while(i<lengths_count-1);</pre>
40
        return "Possible";
41
42
```

	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	~