Qualification Round 2021 - Code Jam 2021

1

SWITCH TO EDITOR

Reversort is an algorithm to sort a list of distinct integers in increasing order. The algorithm is b
"Reverse" operation. Each application of this operation reverses the order of some contiguous p

The pseudocode of the algorithm is the following:

```
Reversort(L):
  for i := 1 to length(L) - 1
    j := position with the minimum value in L between i and length(L), inclusive
    Reverse(L[i..j])
```

After $i-1$ iterations, the positions $1,\ 2,\ \ldots,\ i-1$ of the list contain the $i-1$ smallest elements of L, in increasing order. During the $i$-th iteration, the process reverses the sublist going from the $i$-th position to the current position of the $i$-th minimum element. That makes the $i$-th minimum element end up in the $i$-th position.

For example, for a list with $4$ elements, the algorithm would perform $3$ iterations. Here is how it would process $L = [4, 2, 1, 3]$:

1. $i = 1,\ j = 3 \longrightarrow L = [1, 2, 4, 3]$
2. $i = 2,\ j = 2 \longrightarrow L = [1, 2, 4, 3]$
3. $i = 3,\ j = 4 \longrightarrow L = [1, 2, 3, 4]$

The most expensive part of executing the algorithm on our architecture is the Reverse operation. Therefore, our measure for the cost of each iteration is simply the length of the sublist passed to Reverse, that is, the value $j - i + 1$. The cost of the whole algorithm is the sum of the costs of each iteration.

In the example above, the iterations cost $3$, $1$, and $2$, in that order, for a total of $6$.

You are given a size $\mathbf{N}$ and a cost $\mathbf{C}$. Find a list of $\mathbf{N}$ distinct integers between 1 and $\mathbf{N}$ such that the cost of applying Reversort to it is exactly $\mathbf{C}$, or say that there is no such list.