Code Jam 2021 - Round 2

Minimum Sort

Problem

In this problem, you need to sort a list of $\mathbf{N} = 100$ distinct integers in strictly increasing order. You can rearrange the list by swapping the contents of any two positions (they do not need to be adjacent). Unfortunately, you cannot read those contents directly. You can access information about the list contents by querying the minimum of a range. The minimum query gives you the *position* of the minimum value over a range of consecutive positions. For example, in the list $[51, 33, 100, 11]$, the minimum over the range between positions 2 and 4, inclusive (1-based), is at position 4 and the minimum between positions 1 and 3 is at position 2.

Queries about the minimum within a range are limited by a coin budget per test case. Larger ranges are cheaper: asking about the position of the minimum between positions i and j (for $i < j$) costs $\lceil 10^8/(j - i + 1) \rceil$ coins, where $\lceil x \rceil$ is the smallest integer greater than or equal to x (that is, x rounded up). Swap operations, on the other hand, do not cost any coins.

Write a program that sorts lists of integers using any number of swaps and at most 6×10^8 coins per test case distributed among any number of minimum queries.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, the judge will send you a single line containing two integers $\mathbf{T}$ and $\mathbf{N}$: the number of test cases and the number of elements to sort within each test case, respectively. The judge has the initial lists preset before it gets any input from your program, and the only changes done to them during the exchanges with your program are the swaps that you request.

Then, you must process $\mathbf{T}$ test cases. Each test case consists of a series of exchanges plus an additional line indicating you are done. Each exchange consists of you printing one line and the judge printing one line in response. Your program must print a single line containing one of these options:

- An uppercase `M` and two integers i and j with $i < j$ representing a minimum query. The judge responds with a single integer representing the position of the minimum value in the list within 1-based positions i and j, inclusive.
- An uppercase `S` and two integers i and j with $i < j$ representing a swap operation. The judge swaps the two elements at 1-based positions i and j and responds with `1`.
- An uppercase `D` representing that you are done sorting the list. The judge checks the list. It responds with `1` if the list is sorted in strictly increasing order and `-1` if it is not.

After the judge responds `1` to a `D`, it will finish if it was the last test case or it will immediately start waiting for your first command for the next test case. After receiving the judge's response for the $\mathbf{T}$-th case, your program must finish in order to not receive a Time Limit Exceeded error.

If the judge receives an invalidly formatted line or invalid values from your program at any moment, including a minimum operation whose cost would exceed your remaining budget for the test case, the judge will print a single number `-1`. After the judge prints `-1` for any of the reasons explained above, it will not print any further output. If your program continues to wait for the judge after receiving a `-1`, your program will time out, resulting in a Time Limit Exceeded

error. Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error. As usual, if the memory limit is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

Limits

Time limit: 60 seconds.
Memory limit: 1 GB.

Test Set 1 (Visible Verdict)

$\mathbf{T} = 100$.
$\mathbf{N} = 100$.

Sample Interaction

Judge **Solution**

Constraints

`2 4`
Judge provides $\mathbf{T}$, $\mathbf{N}$

Case 1
List: $[51, 33, 100, 11]$

`M 2 4`
Solution queries for the minimum in the range $[2, 4]$
This operation cost $\lceil 10^8/3 \rceil = 33333334$ *coins*

`4`
Judge provides the position of the minimum in the range

`M 1 3`
Solution queries for the minimum in the range $[1, 3]$
This operation cost $\lceil 10^8/3 \rceil = 33333334$ *coins*

`2`
Judge provides the position of the minimum in the range

`S 1 4`
Solution asks to swap the elements at positions 1 and 4

`1`
Judge responds 1 to a swap operation

The list is now $[11, 33, 100, 51]$

`M 3 4`
Solution queries for the minimum in the range $[3, 4]$
This operation cost $\lceil 10^8/2 \rceil = 50000000$ *coins*

`4`
Judge provides the position of the minimum in the range

`S 3 4`

Solution asks to swap the elements at positions 3 and 4

```
1
```

Judge responds 1 to a swap operation

The list is now $[11, 33, 51, 100]$

```
D
```

Solution tells the judge it is done sorting
The solution spent a total of 116666668 *coins*

```
1
```

Judge responds with 1 saying the list is properly sorted

Case 2
List: $[30, 20, 10, 40]$

```
M 1 4
```

Solution queries for the minimum in the range $[1, 4]$
This operation cost $\lceil 10^8/4 \rceil = 25000000$ *coins*

```
3
```

Judge provides the position of the minimum in the range

```
S 1 3
```

Solution asks to swap the elements at positions 1 and 3

```
1
```

Judge responds 1 to a swap operation

The list is now $[10, 20, 30, 40]$

```
M 3 4
```

Solution queries for the minimum in the range $[3, 4]$
This operation cost $\lceil 10^8/2 \rceil = 50000000$ *coins*

```
3
```

Judge provides the position of the minimum in the range

```
M 2 4
```

Solution queries for the minimum in the range $[2, 4]$
This operation cost $\lceil 10^8/3 \rceil = 33333334$ *coins*

```
2
```

Judge provides the position of the minimum in the range

```
D
```

Solution tells the judge it is done sorting
The solution spent a total of 108333334 *coins*

```
1
```

Judge responds with 1 saying the list is properly sorted

Both Cases Finished

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool