Final Exam - Practice

Session Given The problems are open for submissions from 5pm on Sunday May 10 to 11:59pm on Thursday May 14. You should give yourself 1hr 50 minutes to complete the practice exam. During the days that the problem submissions are open, please do not discuss the problems or their solutions on Piazza.

Starting on May 15, you can still submit the practice problems. But, feel free to also discuss and share your solutions through Piazza.

This exam is:

- open notes/books/any printed resources
- open laptop (anything that you have on your own computer, not online storage)

Online resources that you are allowed to access:

- Gradescope
- course website
- language documentation:
    - Java: https://docs.oracle.com/javase/10/docs/api/
    - C++: https://www.cplusplus.com/reference/ and/or https://en.cppreference.com/w/

Instructions:

Solve three out of the four problems given on the next pages. You will **not** get extra credit for solving all four problems. If you do attempt four problems, we will pick the three with highest scores. For each problem, your last submission counts.

Grading:

Every exam problem is graded out of 10 points. The total exam grade is the weighted sum computed as follows (assume *scoreN* is a score for a particular problem with *score1* >= *score2* >= *score3*):

*exam = 5 * score1 + 3 * score2 + 2 * score3*

The total score for a problem is determined by the maximum between zero and the sum of scores for individual tests based on their results. The maximum score for each test is determined by *max_score = 10/number_of_tests*.

test outcome	test score
passed test	*max_score*
wrong answer	*- 0.5 max_score*
runtime error	*- 0.5 max_score*
timeout error	*- 0.5 max_score*
presentation error	*0.75 max_score*

Jill’s Bicycle

Jill likes to ride her bicycle around the city. But in the city of Carville where she lives some of the streets are not most friendly to the bicyclists. Over the years the bicycling club has rated all the streets’ *safety* on an integer scale: positive values indicate that the street is safe for a person on a bike, negative values indicate that it is unsafe to ride a bike on that street. Jill wants to maximize the safety score along the streets that she is riding her bike. For other parts of her trip, she will just take a bus.

Input

The first line of input contains an integer, N, the number of streets along the route that Jill needs to take, 2 <= N<= 20,000 Each of the next N lines contains a single integer. The i-th integer indicating the safety of the street i. The absolute value of safety for each road will not exceed 10^9.

Output

The program should identify the maximum positive safety score for a given route and print a line containing that value.

If a non-negative score is not possible, the program should print
"No safe streets along this route."

Example 1

```
Input
2
-2
5

Output
5
```

Example 2

```
Input
4
-1
0
-1
-1

Output
No safe streets along this route.
```

Example 3

```
Input
7
-1
3
1
2
-2
5
1

Output
6
```

Toy Blocks

Ayu is playing with toy blocks. Ayu decides to build two towers with those blocks. She wants to use up all of the blocks she has and the number of blocks used in two towers should not differ by more than one. Besides, every block has a height and she wants to minimize the height difference between two towers.

Input

The first line of the input contains one integer N (1 <= N <= 100), the number of toy blocks. Each of the following N lines contains one integer indicating the height h (1 <= h <= 450) of that block.

Output

Print one line, containing two space separated integers, the heights of two towers. The smaller number goes first.

Example 1

```
Input:
3
100
90
200

Output:
190 200
```

Dejavu Center

We call a positive integer a dejavu number if and only if it is only evenly divisible by 1 and itself (which is slightly different from prime numbers since 1 is dejavu but not prime).

Given integer N and C, let L be the list of dejavu numbers between 1 and N and the dejavu center is

- the center part of L with length of 2C if |L| is even and 2C <= |L|
- the center part of L with length of 2C-1 if |L| is odd and 2C-1 <= |L|
- L itself

For example, the dejavu center of N=10, C=2 is {2,3,5} (note L={1,2,3,5,7}). The dejavu center of N=11, C=2 is {2,3,5,7} (note L={1,2,3,5,7,11})

Input

The input consists of a single line, containing two integers N (1 <= N <= 1000) and C (1 <= C <= N).

Output

Print one line `N C: dejavu-center` , where N and C is the input and `dejavu-center` is a space separated integer list representing the dejavu center.

Example 1

```
Input:
21 2

Output:
21 2: 5 7 11
```

Example 2

```
Input:
18 2

Output:
18 2: 3 5 7 11
```

Example 3

```
Input:
18 18

Output:
18 18: 1 2 3 5 7 11 13 17
```

Example 4

```
Input:
100 7

Output:
100 7: 13 17 19 23 29 31 37 41 43 47 53 59 61 67
```

It’s in The Bag

Shirai Kuroko is moving out of her dormitory at the end of the semester. This is a big problem for Kuroko since she has way too many handbags.

As a *teleporter*, she could use teleportation to transfer her bags back home. However, consecutive spatial manipulation is still very tiring for her. She needs to minimize the number of teleportation trip to get all her bags back home.

A bag can be packed within another empty bag if its size is strictly smaller than the outside one. And Kuroko can teleport one piece (the outside bag and all bags inside it) each time she makes a single trip.

For example, given 4 bags with sizes of 4, 3, 2, 1, she can pack it up as shown below

```
|               |
| |           | |
| | |       | | |
| | | |_1_| | | |
| | |___2___| | |
| |_____3_____| |
|_______4_______|
```

(i.e., bag 1 goes into bag2, bag 2 goes into bag 3 and finally bag 3 goes into bag 4) and can teleport only once.

Find the minimum number of teleportation trips necessary for Kuroko to teleport all bags back home.

While maintaining the minimal number of teleportation, it will be hard to transfer too many bags in one teleportation, you are also to minimize the total number of bags in any one piece that must be carried.

Kuroko treasures her bags very much, so she won’t pack a bag into another bag if it already contains another bag in it. That is, two bags with sizes 1 and 2 cannot be placed into a bag of size 4.

Input

The first line contains 1 integer n, the number of bags Kuroko has. 1<=n<=10000.

In the next line, there are n integer smaller than 10^9 indicating the size of bags.

Output

Output one integer K, indicating the minimum number of teleportation trips necessary, followed by K lines, each containing the details about bags for each trip.

The 1st integer m_i of those K lines will be the number of bags transported in that trip, then the following m_i integers should indicate the sizes of the bags in that trip in the sorted order from smallest to largest.

Each size in the input should appear exactly once in the output, and the bags in each piece must fit nested one within another.

If there is more than one solution, any will do.

Example 1

```
Input:
4
1 2 3 4

Output:
1
4 1 2 3 4
```

Example 2

```
Input:
7
1 1 2 2 2 3 3

Output:
3
2 1 2
2 2 3
2 1 2
1 3
```