Koala Game

Koala has created a new game and she has challenged you! She begins by placing down N items, numbered from 0 to $N-1$. She then secretly assigns each item an integer value between 1 and N so that **no two items have been assigned the same value**. Item i has been assigned the value P_i. She has challenged you to identify some properties of the sequence of values $P = P_0, P_1, \ldots, P_{N-1}$.

To do this, you will ask Koala to play a series of *rounds*. In each round, you are given W blue stones, and Koala is given W red stones. You go first by placing some (or possibly all) of your stones next to some items of your choosing. Koala, seeing your arrangement, responds similarly by placing some (or possibly all) of her stones next to some of the items. Koala then wins all items which have **strictly more** red stones than blue stones next to them. Koala always distributes her stones so that she maximises the sum of the values of the items that she wins. If there are are multiple ways to do this, she picks a way which maximises the total number of items that she wins. If there are still multiple ways to do this, she picks any of these ways.

Koala is very lazy and will fall asleep if you ask her to play too many rounds. Your task is to identify patterns in Koala's sequence P by playing as few rounds as possible.

Task

In this task, there are four functions for you to implement: `minValue`, `maxValue`, `greaterValue` and `allValues`. Each function requires you to identify a different property of the sequence P. You are **strongly recommended** to use the template implementation for your language as a starting point for your solution. Note that even if you are only attempting some of the subtasks, you **must** still provide an implementation for all four functions (though some of these implementations may be empty). Your program must not read from standard input, write to standard output or interact with any files.

In each function the parameter `N` is the number of items in the game and the parameter `W` is the number of stones both you and Koala play with in each round of the game.

- `minValue(N, W)` --- This function should return the item number i with the minimum value, that is, $P_i = 1$.
- `maxValue(N, W)` --- This function should return the item number i with the maximum value, that is, $P_i = N$.
- `greaterValue(N, W)` --- This function should compare the value of items 0 and 1, and return the number of the item which is greater. Specifically, it should return `0` if $P_0 > P_1$ and return `1` otherwise.
- `allValues(N, W, P)` --- This function should determine the entire sequence P and place it in the provided array `P`: specifically, `P[i]` should contain the value P_i of item i for all $0 \leq i \leq N-1$.

In each testcase, the grader will call **precisely one** of these functions **one or more times**. Each function call is to be treated as a separate game. Which function is called and the maximum number of times it may be called depends on the subtask (see below). You may assume that Koala has fixed her sequence P before each function call, and it will not change throughout the duration of the function. She may then change her sequence before the next function call.

The implementation for each function should call the function `playRound` to gain information about Koala's sequence.

- `playRound(B, R)` --- Ask Koala to play a round with you.
  - The array `B` describes how many blue stones you place next to each item. Specifically, for all $0 \leq i \leq N-1$, `B[i]` blue stones will be placed next to the item numbered i. Each `B[i]` must be a non-negative integer and the sum `B[0] + B[1] + ... + B[N-1]` must not exceed W.
  - The grader will fill the provided array `R` to describe Koala's response. Specifically, for all $0 \leq i \leq N-1$, Koala will place `R[i]` red stones next to the item numbered i.
  - Each subtask specifies a hard limit on the number of times you may call `playRound` **per game**. Please note that using fewer calls than this limit may yield a higher scoring solution (see below).

Sample Data [0 points]

- There are 5 "Sample Data" testcases. Each testcase calls one of the 4 functions precisely once. See *Examples* below for a detailed description of each testcase.
- $N = 6$.
- $P = 5, 3, 2, 1, 6, 4$.
- You may call `playRound` at most 3200 times per game.

Subtask 1 [4 points]

- In this subtask, the grader will only call the function `minValue`. This function will be called at most 100 times per testcase.
- $N = 100$.
- $W = 100$.
- You may call `playRound` at most 2 times per game.

Subtask 2 [up to 15 points]

- In this subtask, the grader will only call the function `maxValue`. This function will be called at most 100 times per testcase.
- $N = 100$.
- $W = 100$.
- You may call `playRound` at most 13 times per game.

- The score for a testcase in this subtask depends on the maximum number of times $C_{\max}$ that `playRound` is called among all games in that testcase. Precisely your score will be:
  - 15 points if $C_{\max} \leq 4$;
  - 7 points if $5 \leq C_{\max} \leq 13$.

Subtask 3 [up to 18 points]

- In this subtask, the grader will only call the function `greaterValue`. This function will be called at most 1100 times per testcase.
- $N = 100$.
- $W = 100$.
- You may call `playRound` at most 14 times per game.
- The score for a testcase in this subtask depends on the maximum number of times $C_{\max}$ that `playRound` is called among all games in that testcase. Precisely your score will be:
  - 18 points if $C_{\max} \leq 3$;
  - 14 points if $C_{\max} = 4$;
  - 11 points if $C_{\max} = 5$;
  - 5 points if $6 \leq C_{\max} \leq 14$.

Subtask 4 [10 points]

- In this subtask, the grader will only call the function `allValues`. This function will be called **exactly once** per testcase.
- $N = 100$.
- $W = 200$.
- You may call `playRound` at most 700 times.

Subtask 5 [up to 53 points]

- In this subtask, the grader will only call the function `allValues`. This function will be called **exactly once** per testcase.
- $N = 100$.
- $W = 100$.
- You may call `playRound` at most 3200 times.
- The score for a testcase in this subtask depends on the number of times C that `playRound` is called. Precisely your score will be:
  - 53 points if $C \leq 100$;
  - $\lfloor 53 - 8 \log_2 (C/100) \rfloor$ points if $101 \leq C \leq 3200$, where $\lfloor x \rfloor$ is the greatest integer less than or equal to x. In particular, if $C = 3200$ your solution will score 13 points.

Scoring

- In each testcase, your program must always run within the time and memory limits for this task. **This includes the time and memory consumed by the grader** when setting up, tearing down and responding to calls to the function `playRound`. When estimating this overhead you may assume that the grader used in judging has **identical functionality** and **similar implementation** to the sample grader provided.
- If `playRound` is called with an invalid array `B`, or the number of calls to `playRound` exceeds the hard limit for any game within a testcase, the entire testcase will be marked as `Not Correct`, scoring 0.
- If a function does not correctly determine the required properties of Koala's sequence P for a particular game within a testcase, the entire testcase will be marked as `Not Correct`, scoring 0.
- Both Subtask 4 and Subtask 5 require you to implement the function `allValues`, with different values of W. You may use this to differentiate the two subtasks in your implementation -- see the template implementation in your language for further details.
- You may make a maximum of **60** submissions for this task with a minimum interval of 2 minutes between successive submissions.

Examples

Consider the following sequence P.

i	0	1	2	3	4	5
P_i	5	3	2	1	6	4

Below are a series of example calls made to `playRound`, and a valid response by the grader to each (note that there may be more than one valid response for any given call to `playRound`).

W	Sample function call	Possible grader response	Explanation
6	`playRound([0, 3, 0, 2, 1, 0], R)`	`R = [1, 1, 1, 0, 2, 1]`	You place three, two and one blue stone(s) next to items 1, 3 and 4, respectively, and no stones next to items 0, 2 and 5. In reply, Koala places a single red stone next to items 0, 1, 2 and 5 and two red stones next to item 4 with no red stones next to item 3. Thus, she wins items 0, 2, 4 and 5 with a total combined value of $5+2+6+4=17$, which is the maximum possible.
6	`playRound([1, 2, 3, 1, 2, 0], R)`	Invalid function call. Your program is terminated and marked `Not Correct`, scoring `0` for this testcase.	You have placed $1+2+3+1+2=9>6=W$ stones, which is invalid.

W	Sample function call	Possible grader response	Explanation
12	`playRound([1, 2, 3, 1, 2, 0], R)`	`R = [2, 3, 0, 2, 3, 1]`	You do not need to place all W of your blue stones, and Koala does not need to place all W of her red stones.
6	`playRound([0, 1, 0, 0, 1, 0], R)`	`R = [1, 0, 1, 1, 2, 1]`	If there are multiple possible responses for Koala that maximise her total value, she chooses one that maximises the total number of items that she wins, hence `R = [1, 2, 0, 0, 2, 1]` is not a valid response.

Feedback for each of the functions that the grader calls below (exactly one per testcase) is provided in the given order as "Sample Data" upon submission. You may call `playRound` at most 3200 times in each of these 5 testcases.

#	Grader calls	Expected return value	Explanation
1	`minValue(6, 6)`	`3`	$P_3 = 1$, so item 3 has the minimum value.
2	`maxValue(6, 6)`	`4`	$P_4 = 6 = N$, so item 4 has the maximum value.
3	`greaterValue(6, 6)`	`0`	$P_0 = 5 > 3 = P_1$, so item 0 has a greater value than item 1.
4	`allValues(6, 12, P)`	None, `P = [5, 3, 2, 1, 6, 4]`	The function `allValues` has no return value, but instead places the correct values in the given array `P`.
5	`allValues(6, 6, P)`	None, `P = [5, 3, 2, 1, 6, 4]`	Same as the previous function call.

Sample Grader

The sample grader reads from standard input in the following format:

- line 1: two integers F, G;
- lines 2 to $G + 1$: each line contains two integers N, W followed by N integers $P_0, P_1, \ldots, P_{N-1}$ describing a single game.

The integer F determines which function the sample grader will call:

F	Function called
1	`minValue`
2	`maxValue`
3	`greaterValue`
4	`allValues`

The integer G determines how many times the specified function will be called. Each subsequent line describes a game with Koala's sequence.

The sample grader will write two lines to standard output for each function call. The first line contains

the number of times playRound was called.

- If $F = 4$, the second line contains the contents placed in the array P by your implementation of the function allValues;
- Otherwise, if $F = 1, 2$ or 3, the second line contains a single integer: the return value of the corresponding function.

For instance, the fourth testcase in "Sample Data" (which calls allValues with $N = 6$ and $W = 12$) can be described with the following sample input file.

```
4 1
6 12 5 3 2 1 6 4
```