코알라

코알라가 새로운 게임을 만들어서 당신에게 도전했다! 코알라는 먼저 N개의 아이템을 가지고 시작하는데, 이 아이템은 0에서 $N-1$까지 번호가 매겨져 있다. 다음 코알라는 각 아이템에 1부터 N까지의 정수로 값을 몰래 주었는데, **각 아이템은 서로 다른 값을 가진다**. i번 아이템의 값은 P_i이다. 다음 코알라는 값들의 서열 $P = P_0, P_1, \ldots, P_{N-1}$의 특징을 당신이 알아내게 한다.

이를 위해서, 당신은 일련의 *라운드*로 이루어진 다음 게임을 코알라와 한다. 각 라운드마다, 당신은 W개의 파란 돌, 코알라는 W개의 붉은 돌을 가지고 시작한다. 당신이 먼저 아이템들을 고른 뒤 아이템 옆에 당신이 가진 돌의 일부 (또는 전부)를 놓는다. 코알라는 당신의 돌 배치를 보고, 비슷한 방식으로 아이템들을 고른 뒤 아이템 옆에 코알라가 가진 돌의 일부 (또는 전부)를 놓는다. 붉은 돌이 파란 돌보다 **많이** 놓인 아이템은 코알라가 가지게 된다. 코알라는 항상 코알라가 갖게 되는 아이템들의 값이 최대가 되도록 붉은 돌을 배치한다. 만약 이렇게 돌을 배치하는 방법이 둘 이상 있다면, 코알라는 얻는 아이템의 수가 최대가 되는 방법을 택한다. 그래도 둘 이상의 방법이 있다면, 그때는 코알라는 이 중 아무거나 하나를 택한다.

코알라는 아주 게을러서, 너무 많은 라운드를 돌면 잠들어버린다. 당신의 임무는 가능한 한 적은 수의 라운드를 돌면서 코알라의 서열 P의 패턴을 알아내는 것이다.

태스크

이 문제에서 당신은 4개의 함수를 구현해야 한다: `minValue`, `maxValue`, `greaterValue`, `allValues`. 각 함수마다 당신은 서열 P의 서로 다른 속성을 구해야 한다. 해법을 구하기 위해서 당신이 사용하는 프로그래밍 언어로 쓰여진 템플리트에서 시작할 것을 **강력히 권장한다**. 만약 서브태스크 중 일부만을 풀려고 하더라도, 네 함수 모두를 구현해서 제출해야 **한다** (비록 이 중 일부의 구현이 비어 있더라도).당신의 프로그램은 표준 입력에서 읽거나, 표준 출력으로 출력하거나, 다른 파일과 상호소통을 하면 절대로 안된다.

각 함수에서 파라미터 `N`은 게임에서 사용하는 아이템의 수이고, 파라미터 `W`는 당신과 코알라가 게임에 사용하는 돌의 개수이다.

- `minValue(N, W)` --- 이 함수는 가장 작은 값을 갖는 아이템 번호 i를 리턴해야 한다. 즉, $P_i = 1$이다.
- `maxValue(N, W)` --- 이 함수는 가장 큰 값을 갖는 아이템 번호 i를 리턴해야 한다. 즉, $P_i = N$이다.
- `greaterValue(N, W)` --- 이 함수는 아이템 0과 1의 값을 비교한 후, 큰 값을 갖는 아이템의 번호를 리턴한다. 구체적으로, 만약 $P_0 > P_1$ 이면 리턴값은 `0`이고 그렇지 않으면 리턴값은 `1`이다.
- `allValues(N, W, P)` --- 이 함수는 서열 P의 전체값을 알아낸 뒤 주어진 배열 `P`에 저장한다. 구체적으로, `P[i]`의 값은 모든 $0 \le i \le N-1$에 대해서 아이템 i의 값 P_i이어야 한다.

각각의 테스트케이스에서, 그레이더는 이 함수중 **정확히 하나**를 **한번 또는 그 이상** 호출한다. 각 함수 호출 하나하나는 별도의 게임으로 간주된다. 어떤 함수가 호출되며, 최대 몇번 호출되는지는

아래의 서브태스크에 따라 정해진다. 코알라는 매번 함수를 호출하기 전 서열 P를 결정했고, 함수가 호출되는 동안 서열이 바뀌지 않는다고 봐도 좋다. 다음번 함수가 호출되기 전 코알라는 서열을 바꿀 수 있다.

각각의 함수를 구현하기 위해서, 코알라의 서열 정보를 얻으려면 함수 playRound를 호출해야 한다.

- `playRound(B, R)` --- 코알라가 당신과 한 라운드를 하도록 부탁한다.
    - 배열 `B`는 당신이 각각의 아이템 옆에 몇개의 파란 돌을 놓았는지를 기술한다. 구체적으로, 모든 $0 \leq i \leq N-1$에 대해서, 당신은 `B[i]`개의 푸른 돌을 아이템 i 옆에 놓았다. 각각의 `B[i]` 값은 음이 아닌 정수이며, 이들의 합 `B[0] + B[1] + ... + B[N-1]`은 W를 초과해서는 안된다.
    - 그레이더는 주어진 배열 `R`에 코알라의 반응을 저장하여 보내준다. 구체적으로, 모든 $0 \leq i \leq N-1$에 대해서, 코알라는 `R[i]` 개의 붉은 돌을 아이템 i 옆에 놓는다.
    - 각 서브태스크는 **매 게임마다** 당신이 playRound 함수를 최대 몇 번 호출할 수 있는지 엄격한 제한을 두었다. 이 제한보다 적은 수로 함수를 호출하면 점수를 더 많이 받을 수 있다 (아래를 참조하시오).

예제 데이터 [0 점]

- 5개의 "예제 데이터" 테스트케이스가 주어진다. 각 테스트케이스는 4개의 함수 중 하나를 정확하게 한번 호출한다. 아래의 *예제*를 참고하여 각 테스트케이스 별로 자세한 설명을 보시오.
- $N = 6$.
- $P = 5, 3, 2, 1, 6, 4$.
- playRound를 한 게임에 최대 3200번 호출할 수 있다.

서브태스크 1 [4 점]

- 이 서브태스크에서 그레이더는 함수 `minValue`만 호출한다. 이 함수는 테스트케이스마다 최대 100번 호출된다.
- $N = 100$.
- $W = 100$.
- playRound를 한 게임에 최대 2번 호출할 수 있다.

서브태스크 2 [최대 15 점]

- 이 서브태스크에서 그레이더는 함수 `maxValue`만 호출한다. 이 함수는 테스트케이스마다 최대 100번 호출된다.
- $N = 100$.
- $W = 100$.
- playRound를 한 게임에 최대 13번 호출할 수 있다.
- 이 서브태스크에서 한 테스트케이스의 점수는 이 테스트케이스에 속한 모든 게임에

playRound 함수를 호출한 회수의 최대값 $C_{\max}$에 의해서 결정된다. 구체적으로, 당신의 점수는 다음과 같다.

- 만약 $C_{\max} \leq 4$이면 15점;
- 만약 $5 \leq C_{\max} \leq 13$이면 7점.

서브태스크 3 [최대 18 점]

- 이 서브태스크에서 그레이더는 함수 greaterValue만 호출한다. 이 함수는 테스트케이스마다 최대 1100번 호출된다.
- $N = 100$.
- $W = 100$.
- playRound를 한 게임에 최대 14번 호출할 수 있다.
- 이 서브태스크에서 한 테스트케이스의 점수는 이 테스트케이스에 속한 모든 게임에 playRound 함수를 호출한 회수의 최대값 $C_{\max}$에 의해서 결정된다. 구체적으로, 당신의 점수는 다음과 같다.
  - 만약 $C_{\max} \leq 3$이면 18점;
  - 만약 $C_{\max} = 4$이면 14점;
  - 만약 $C_{\max} = 5$이면 11점;
  - 만약 $6 \leq C_{\max} \leq 14$이면 5점.

서브태스크 4 [10 점]

- 이 서브태스크에서 그레이더는 함수 allValues만 호출한다. 이 함수는 테스트케이스마다 **정확히 한번** 호출된다.
- $N = 100$.
- $W = 200$.
- playRound를 최대 700번 호출할 수 있다.

서브태스크 5 [최대 53 점]

- 이 서브태스크에서 그레이더는 함수 allValues만 호출한다. 이 함수는 테스트케이스마다 **정확히 한번** 호출된다.
- $N = 100$.
- $W = 100$.
- playRound를 최대 3200번 호출할 수 있다.
- 이 서브태스크에서 한 테스트케이스의 점수는 playRound 함수를 호출한 회수의 최대값 $C_{\max}$에 의해서 결정된다. 구체적으로, 당신의 점수는 다음과 같다.
  - 만약 $C \leq 100$이면 53점;
  - 만약 $101 \leq C \leq 3200$이면 $\lfloor 53 - 8 \log_2 (C/100) \rfloor$ 점인데, $\lfloor x \rfloor$는 x보다 같거나 작

은 가장 큰 정수이다. 특히, 만약 $C = 3200$이면 13점.

점수

- 각 테스트케이스마다 당신의 프로그램은 시간과 메모리 제한 이내에서 동작해야 한다. **이는 그레이더가 쓰는 시간과 메모리까지 포함된다.** 즉, 데이터를 세팅하고 이를 처리하여 함수 playRound의 호출에 응답하는 시간과 메모리를 포함한다. 이 오버헤드를 예측하기 위해서, 채점에 사용되는 그레이더는 제공되는 샘플 그레이더와 **동일한 기능**을 가지고 **비슷하게 구현**되어 있다고 가정해도 좋다.
- 만약 playRound 함수가 잘못된 배열 B를 가지고 호출되었거나, playRound 함수 호출 횟수가 테스트케이스에서 정해진 엄격한 제한을 넘어섰다면, 전체 테스트 케이스는 `Not Correct`로 채점되어 0점을 얻게 된다.
- 만약 테스트케이스 내의 어떤 게임에 대해서, 당신이 구현한 함수가 코알라의 서열 P에 대해 요구한 특징을 정확하게 구하지 못했다면, 전체 테스트케이스는 `Not Correct`로 채점되어 0점을 얻게 된다.
- 서브태스크 4와 5는 모두 당신이 W의 다른 값에 대해 allValues를 구현할 것을 요구한다. 이를 이용하여, 구현시 두 서브태스크를 구별할 수 있다. 보다 상세한 내용은 당신이 사용하는 프로그래밍 언어로 된 템플리트를 참고하라.
- 이 문제는 최대 60번 제출할 수 있고, 한번 제출한 다음에는 최소 2분이 지나야 다음 제출을 할 수 있다.

예제

다음 서열 P를 생각해보자.

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

아래에는 playRound 함수에 대해서 여러번 예제 호출한 것과, 각각에 대한 그레이더의 타당한 응답이 주어져 있다. (어떤 playRound 함수의 호출에 대해서는 둘 이상의 타당한 응답이 가능하다는데 주의하시오)

W	예제 함수 호출	가능한 그레이더 응답	설명
6	`playRound([0, 3, 0, 2, 1, 0], R)`	`R = [1, 1, 1, 0, 2, 1]`	당신은 1, 3, 4번 아이템 옆에 각각 3, 2, 1개의 푸른 돌을 놓았고, 0, 2, 5번 아이템 옆에는 아무 돌도 놓지 않았다. 그에 대한 응답으로 코알라는 붉은 돌 하나씩을 0, 1, 2번 아이템 옆에 놓았고, 2개의 붉은 돌을 4번 아이템 옆에 놓고, 3번 아이템 옆에는 아무 돌도 놓지 않았다. 코알라는 0, 2, 4, 5번 아이템을 가지게 되었으며 값의 총합은 $5 + 2 + 6 + 4 = 17$으로 최대값이 된다.

W	예제 함수 호출	가능한 그레이더 응답	설명
6	`playRound([1, 2, 3, 1, 2, 0], R)`	잘못된 함수 호출이다. 당신의 프로그램은 종료되며 결과는 `Not Correct`로, 이 테스트케이스의 점수는 `0`점이다.	당신은 총 $1+2+3+1+2=9>6=W$ 개의 돌을 놓았는데, 이는 잘못되었다.
12	`playRound([1, 2, 3, 1, 2, 0], R)`	`R = [2, 3, 0, 2, 3, 1]`	당신이 모든 W개의 푸른 돌을 놓을 필요는 없으며, 코알라도 모든 W개의 붉은 돌을 놓을 필요는 없다.
6	`playRound([0, 1, 0, 0, 1, 0], R)`	`R = [1, 0, 1, 1, 2, 1]`	만약 코알라가 값의 합을 최대로 하도록 하는 둘 이상의 대응 방법이 있다면, 갖는 아이템의 수가 최대가 되는 쪽을 택한다. 따라서 `R = [1, 2, 0, 0, 2, 1]` 은 타당한 답이 아니다.

그레이더가 아래와 같이 호출하는 함수 (테스트케이스마다 정확히 하나) 각각에 대한 응답은 제출 시 순서대로 "Sample Data"로 제공된다.
다음 5개의 테스트케이스 각각에 대해서 playRound 함수를 최대 3200번 호출할 수 있다.

#	그레이더 호출	예측되는 리턴 값	설명
1	`minValue(6, 6)`	`3`	$P_3=1$이므로 3번 아이템이 가장 작은 값.
2	`maxValue(6, 6)`	`4`	$P_4=6=N$이므로 4번 아이템이 가장 큰 값.
3	`greaterValue(6, 6)`	`0`	$P_0=5>3=P_1$ 이므로 0번 아이템이 1번 아이템보다 큰 값.
4	`allValues(6, 12, P)`	None, `P = [5, 3, 2, 1, 6, 4]`	`allValues` 함수는 리턴값이 없는 대신, 정확한 값을 주어진 배열 `P`에 저장한다.
5	`allValues(6, 6, P)`	None, `P = [5, 3, 2, 1, 6, 4]`	위와 동일.

Sample Grader

샘플 그레이더는 입력을 표준 입력으로부터 다음 형식으로 읽는다.

- line 1: 두 정수 F, G;
- lines 2 to $G+1$: 매 줄마다 두 정수 N, W 가 주어진 뒤 게임 하나를 기술하는 N 개의 정수 $P_0, P_1, \ldots, P_{N-1}$.

정수 F는 샘플 그레이더가 어떤 함수를 호출할 지 결정한다.

F	호출 함수
1	`minValue`
2	`maxValue`
3	`greaterValue`

F	호출 함수
4	allValues

정수 G는 정해진 함수를 몇번 호출할 것인지 결정한다. 그 다음에 오는 줄들은 코알라의 서열과 함께 게임에 대한 정보를 기술한다.

샘플 그레이더는 매 함수 호출마다 표준 출력으로 두 줄을 출력한다. 첫줄은 playRound 함수의 호출 횟수이다.

- 만약 $F = 4$이면, 두번째 줄에는 당신의 allValues 구현에 의해 배열 P에 들어간 내용이 출력된다;
- 그 외 $F = 1, 2, 3$인 경우에는, 두번째 줄에는 해당 함수의 리턴값에 해당하는 정수 하나가 출력된다.

예를 들어, "Sample Data"의 네번째 테스트케이스는 (allValues를 $N = 6, W = 12$로 호출) 다음 예제 입력 파일로 기술할 수 있다.

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