Malika taught a new fun time program practice for Engineering Students.

As a part of this she has given set of N numbers, and asked the students

to perform the operations listed below:

First, build the segment tree and then do the following,

1. sumRange(s1, s2) - return the sum of numbers between the indices

s1 and s2, both are inclusive and 0<=s1<=s2<N.

2. update(ind, val) - update the value at the index 'ind' to 'val'.

Your task is to solve this problem using Segment Tree concept.

Input Format:

-------------

Line-1: Two integers N and Q, size of the array(set of numbers) and query count.

Line-2: N space separated integers.

next Q lines: Three integers option, start/ind and end/val.

Output Format:

--------------

An integer result, for every sumRange query.

Sample Input:

-------------

8 5

1 2 13 4 25 16 17 8

1 2 6 //sumRange

1 0 7 //sumRange

2 2 18 //update

2 4 17 //update

1 2 7 //sumRange

Sample Output:

--------------

75

86

80

import java.util.\*;

class Solution{

int[] nums;

int[] BIT;

int n;

public Solution(int[] nums){

this.nums = nums;

n = nums.length;

BIT = new int[n + 1];

for (int i = 0; i < n; i++){

init(i, nums[i]);

}

}

public void init(int i, int val){

i++;

while (i <= n){

BIT[i] += val;

i += (i & -i);

}

}

void update(int i, int val){

int diff = val - nums[i];

nums[i] = val;

init(i, diff);

}

public int getSum(int i){

int sum = 0;

i++;

while (i > 0){

sum += BIT[i];

i -= (i & -i);

}

return sum;

}

public int sumRange(int i, int j){

return getSum(j) - getSum(i - 1);

}

public static void main(String args[] ){

Scanner scan = new Scanner(System.in);

int n=scan.nextInt();

int q=scan.nextInt();

int[] nums=new int[n];

for(int i=0; i<n; i++){

nums[i] = scan.nextInt();

}

Solution ft =new Solution(nums);

while(q-->0){

int opt=scan.nextInt();

if(opt==1){

int s1 = scan.nextInt();

int s2 = scan.nextInt();

System.out.println(ft.sumRange(s1,s2));

}else{

int ind = scan.nextInt();

int val= scan.nextInt();

ft.update(ind,val);

}

}

}

}

A courier company has to deliver N items with different heights.

All the items have to deliver to one building only.

The building has K rooms inside indexed 0,1,...,K-1, and the heights of

the rooms are different. All the rooms connected with each other and has

two entry points, one entry is at frontside and other entry is at backside of

the building . e.g, you can reach room-1 from room-0 only, room-2 from

room-1 only, ..so on, and vice-versa.

Now the task of the delivery agent is to keep the items inside the building.

To keep the items into the rooms, delivery agent has to follow these rules:

- Agent can't keep, two goods in one room.

- Agent can enter into the building either from frontside or from back side,

If the height of some room is less than the height of an item,

then the item will be stopped before that room, so are the items behind it.

- Agent can rearrange the order of items to keep them inside the building.

Your task is to return the maximum number of items, the delivery agent can keep

inside the building.

Input Format:

-------------

Line-1 -> two integers N and K, number of items and number of rooms.

Line-2 -> N space separated integers, heights of the items.

Line-3 -> K space separated integers, heights of the rooms.

Output Format:

--------------

Print an integer as result.

Sample Input-1:

---------------

5 4

2 3 3 4 5

4 5 2 3

Sample Output-1:

----------------

4

Explanation:

------------

We can first keep the item of height 2 in room 2.

Then we can keep the item of height 3 in room 3.

Then we can keep the item of height 3 in room 1.

Then we can keep the item of height 4 in room 0.

Sample Input-2:

---------------

4 5

4 6 6 3

3 2 4 5 6

Sample Output-2:

----------------

3

Explanation:

------------

Notice that it's not possible to keep the two items of height 6 into the building,

since there's only 1 room of height >= 6

Other valid solutions are to put the item with height 4 in room 2 or

to put the item with height 3 in room 2 or room 0 before putting

the the item with height 4 and 6.

import java.util.\*;

public class Main{

public static void main(String args[]){

Scanner sc= new Scanner(System.in);

int n= sc.nextInt();

int k= sc.nextInt();

int items[]= new int[n];

int rooms[]= new int[k];

for(int i=0;i<n;i++){

items[i]= sc.nextInt();

}

for(int i=0;i<k;i++){

rooms[i]= sc.nextInt();

}

Arrays.sort(items);

int low=0;

int high=rooms.length-1;

int ind=n-1;

int count=0;

while(ind>=0 && low<=high){

if(rooms[low]==items[ind]){

low++;

count++;

}

else if(rooms[high]==items[ind]){

high--;

count++;

}

else if(rooms[low]>=items[ind]){

low++;

count++;

}

else if(rooms[high]>=items[ind]){

high--;

count++;

}

ind--;

}

System.out.println(count);

}

}

There are n football players standing in the ground, coach wants to know the

P'th largest height of the players. Given an array of heights[] and the value of P.

Help the coach to find the P'th largest height.

Note: You are suppose to print the P'th largest height in the sorted order of heights[].

Not the P'th distinct height.

Input Format:

-------------

Line-1: Size of array n and P value(space separeted)

Line-2: Array elements of size n.

Output Format:

--------------

Print P'th largest height.

Sample input-1:

---------------

8 2

1 2 1 3 4 5 5 5

Sample output-1:

----------------

5

Sample input-2:

---------------

6 3

2 4 3 1 2 5

Sample output-2:

----------------

3

import java.util.\*;

public class Main{

public static void main(String args[]){

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int p=sc.nextInt();

int[] a=new int[n];

for(int i=0;i<n;i++){

a[i]=sc.nextInt();

}

Arrays.sort(a);

System.out.print(a[n-p]);

}

}