**Maximum product**

Show Topic Tags   

Given an array A[] of n integers, the task is to find a subsequence of size k whose product is maximum among all possible k sized subsequences of given array.

**Input:**  
First line consists of T test cases. First line of every test cases consists of N, denoting the number of elements in an array. Second line consists of elements of array.

**Output:**  
Single line output, print the maximum possible product.

**Constraints:**  
1<=T<=100  
1<=N<=10000  
1<=K<=N  
1<=Ai<=1000

**Example:  
Input:**  
2  
4 3  
8 6 4 6   
7 3  
10 2 3 8 1 10 4   
**Output:**  
288  
800

\*\*For More Examples Use Expected Output\*\*

<http://practice.geeksforgeeks.org/problems/maximum-product/0>

import java.util.\*;

import java.lang.\*;

import java.io.\*;

import java.math.BigInteger;

class GFG {

public static void main(String[] args) throws IOException {

// TODO code application logic here

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t = Integer.parseInt(br.readLine());

while(t-- > 0) {

String[] nk = br.readLine().trim().split(" ");

int n = Integer.parseInt(nk[0]);

int k = Integer.parseInt(nk[1]);

String[] input = br.readLine().trim().split( " ");

int[] arr = new int[n];

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

arr[i] = Integer.parseInt(input[i]);

}

Arrays.sort(arr);

int cont =0;

BigInteger prod = BigInteger.valueOf(1);

for(int i = n-1; cont < k && i>=0; i--) {

//prod\*= arr[i];

prod = prod.multiply(BigInteger.valueOf(arr[i]));

cont++;

}

System.out.println(prod);

}

}

}