**Mega Sale**

Show Topic Tags   

LALU wanted to purchase a laptop so he went to a nearby sale.There were ***n***Laptops at a sale. Laptop with index ***i***costs ***ai*** rupees. Some Laptops have a negative price — their owners are ready to pay LALU if he buys their useless Laptop. LALU can buy any Laptop he wants. Though he's very strong, he can carry at most ***m*** Laptops, and he has no desire to go to the sale for the second time. Please, help LALU find out the maximum sum of money that he can earn.

**Input:**

First line of the input contains**T** denoting the number of test cases.Each test case has 2 lines :

* first line has two spaced integers n m.
* second line has n integers [a0...ai...an-1].

**Output:**

The maximum sum of money that LALU can earn, given that he can carry at most *m* Laptops.

**Constraints:**

1≤**T**≤10

1≤**n,m**≤100

-1000≤**ai**≤1000

**Sample Input:**

1

 5 3

-6 0 35 -2 4

**Sample Output:**

8

**Explanation:**

LALU takes the laptops with -6 and -2 and thus earns 8 rupees.

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

<http://practice.geeksforgeeks.org/problems/mega-sale/0>

/\*

\* To change this template, choose Tools | Templates

\* and open the template in the editor.

\*/

package javaapplication242;

import java.util.\*;

import java.lang.\*;

import java.io.\*;

/\*\*

\*

\* @author Administrador

\*/

public class JavaApplication242 {

/\*\*

\* @param args the command line arguments

\*/

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

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

int m = Integer.parseInt(nm[1]);

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

//String[] input = "-931 -806 -830 -828 -916 -962 -660 -867 -952 -966 -820 -906 -724 -982 -680 -717 -488 -741 -897 -613 -986 -797 -964 -939 -808 -932 -810 -860 -641 -916 -858 -628 -821 -929 -917 -976 -664 -985 -778 -665 -624 -928 -940 -958 -884 -757 -878 -896 -634 -526 -514 -873 -990 -919 -988 -878 -650 -973 -774 -783 -733 -648 -756 -895 -833 -974 -832 -725 -841 -748 -806 -613 -924 -867 -881 -943 -864 -991 -809 -926 -777 -817 -998 -682 -910 -996 -241 -722 -964 -904 -821 -920 -835 -699 -805 -632 -779 -317 -915 -654".trim().split(" ");

int[] arr = new int[n];

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

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

}

Arrays.sort(arr);

int sum =0, i =0;

while(i < arr.length && arr[i] <= 0 && i < m) {

sum += arr[i];

i++;

}

System.out.println(Math.abs( sum));

}

}

}