Power Hungry

============

Commander Lambda's space station is HUGE. And huge space stations take a LOT of power. Huge space stations with doomsday devices take even more power. To help meet the station's power needs, Commander Lambda has installed solar panels on the station's outer surface. But the station sits in the middle of a quasar quantum flux field, which wreaks havoc on the solar panels. You and your team of henchmen have been assigned to repair the solar panels, but you'd rather not take down all of the panels at once if you can help it, since they do help power the space station and all!

You need to figure out which sets of panels in any given array you can take offline to repair while still maintaining the maximum amount of power output per array, and to do THAT, you'll first need to figure out what the maximum output of each array actually is. Write a function solution(xs) that takes a list of integers representing the power output levels of each panel in an array, and returns the maximum product of some non-empty subset of those numbers. So for example, if an array contained panels with power output levels of [2, -3, 1, 0, -5], then the maximum product would be found by taking the subset: xs[0] = 2, xs[1] = -3, xs[4] = -5, giving the product 2\*(-3)\*(-5) = 30. So solution([2,-3,1,0,-5]) will be "30".

Each array of solar panels contains at least 1 and no more than 50 panels, and each panel will have a power output level whose absolute value is no greater than 1000 (some panels are malfunctioning so badly that they're draining energy, but you know a trick with the panels' wave stabilizer that lets you combine two negative-output panels to produce the positive output of the multiple of their power values). The final products may be very large, so give the solution as a string representation of the number.

Languages

=========

To provide a Python solution, edit solution.py

To provide a Java solution, edit Solution.java

Test cases

==========

Your code should pass the following test cases.

Note that it may also be run against hidden test cases not shown here.

-- Python cases --

Input:

solution.solution([2, 0, 2, 2, 0])

Output:

8

Input:

solution.solution([-2, -3, 4, -5])

Output:

60

-- Java cases --

Input:

Solution.solution({2, 0, 2, 2, 0})

Output:

8

Input:

Solution.solution({-2, -3, 4, -5})

Output:

60

Use verify [file] to test your solution and see how it does. When you are finished editing your code, use submit [file] to submit your answer. If your solution passes the test cases, it will be removed from your home folder.