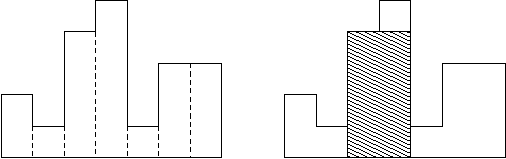
A histogram is a polygon composed of a sequence of rectangles aligned at a common base line. The rectangles have equal widths but may have different heights. For example, the figure on the left shows the histogram that consists of rectangles with the heights 2, 1, 4, 5, 1, 3, 3, measured in units where 1 is the width of the rectangles:



Usually, histograms are used to represent discrete distributions, e.g., the frequencies of characters in texts. Note that the order of the rectangles, i.e., their heights, is important. Calculate the area of the largest rectangle in a histogram that is aligned at the common base line, too. The figure on the right shows the largest aligned rectangle for the depicted histogram.

Input

The input contains several test cases. Each test case describes a histogram and starts with an integer *n*, denoting the number of rectangles it is composed of. You may assume that *1<=n<=100000*. Then follow *n* integers *h1,...,hn*, where *0<=hi<=1000000000*. These numbers denote the heights of the rectangles of the histogram in left-to-right order. The width of each rectangle is *1*. A zero follows the input for the last test case.

Output

For each test case output on a single line the area of the largest rectangle in the specified histogram. Remember that this rectangle must be aligned at the common base line.

Sample Input

7 2 1 4 5 1 3 3

4 1000 1000 1000 1000

0

Sample Output

8

4000

Hint

Huge input, scanf is recommended.

可转化为求找一个子序列。 使得这个序列的长度乘以序列最小数最大。  
分析：这是一个单调栈的问题，维护栈单调不减。 单调栈 主要是大家要自己枚举，需要找到每个元素 最左能扩展到那 ，最右能扩展到那，当然最小的是你枚举的那个元素。

很容易想到的就是使用单调栈。维护一个从栈顶到栈低减小的栈！   
每次遇到栈顶的元素大于要压入的元素就要，出栈然后和当前的最大面积取最大值，这样一直下去，最后栈中就剩下一个从栈顶到栈低减小的栈。最后变为的栈就像下面这个图一样！

#include<cstdio>

#include<stack>

#include<algorithm>

#include<iostream>

typedef long long ll;

using namespace std;

int main()

{

int n,height;

while(scanf("%d",&n)&&n)

{ //当栈顶元素大于当前元素 //就要开始弹栈了

stack<pair<ll,ll> > q;

ll ans(0);

for(int i=1;i<=n;i++)

{

scanf("%d",&height);

ll width=0;

while(!q.empty() && q.top().first>=height)

{

ll H=q.top().first;

ll W=q.top().second;

q.pop();

width+=W;

//每次出栈的时候，

// 就是计算以这个矩形的高度为最长高度的最大面积,

//因为左边的比他小，所以只需要向右延伸就可以了

//所以每次出栈的时候，

//就是出栈后的栈顶元素宽度+1

ans=max(ans,H\*width);//如果大于当前的最大面积就更新

}

q.push(make\_pair((ll)height,(ll)width+1));

}

int tmp(0);

while(q.empty()==0)

{

ans=max(ans,q.top().first\*(tmp+q.top().second));

tmp+=q.top().second;

q.pop();

}

printf("%lld\n",ans);

}

return 0;

}