Largest Rectangle



*Skyline Real Estate Developers* is planning to demolish a number of old, unoccupied buildings and construct a shopping mall in their place. Your task is to find the largest solid area in which the mall can be constructed.

There are a number of buildings in a certain two-dimensional landscape. Each building has a height, given by . If you join adjacent buildings, they will form a solid rectangle of area

.

**Example**

A rectangle of height and length can be constructed within the boundaries. The area formed is .

**Function Description**

Complete the function largestRectangle int the editor below. It should return an integer representing the largest rectangle that can be formed within the bounds of consecutive buildings.

largestRectangle has the following parameter(s):

*int h[n]:* the building heights

**Returns**

- *long:* the area of the largest rectangle that can be formed within the bounds of consecutive buildings

**Input Format**

The first line contains , the number of buildings.

The second line contains space-separated integers, each the height of a building.

**Constraints**

**Sample Input**

STDIN Function

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

5 h[] size n = 5

1 2 3 4 5 h = [1, 2, 3, 4, 5]

**Sample Output**

9

**Explanation**



An illustration of the test case follows.

1 2 3 4 5

**Chạy code:**

Ví dụ: [2, 3, 4, 5, 4] => max\_area = 12

1. **i = 0:** Stack.empty = true => thêm index của 2 (0) vào stack



1. **i = 1:** 3 > 2 => thêm index của 3 (1) vào stack



1. i = 2: 4 > 3 => thêm index của 4 (2) vào stack

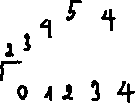


1. **i = 3:** 5 > 4 => thêm index của 5 (3) vào stack



1. **i = 4:** 4 < 5

top = stack.peek = 4



stack.pop => Lúc này stack.peek = 2

h[top] = 5



subarea = h[top] \* (i – stack.peek – 1) = 5 \* (4 – 2 – 1) = 5



max\_area < subarea => max\_area = subarea = 5

vòng lặp tiếp: 4 = 4 => thêm index của 4 (4) vào stack

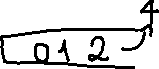


1. **i = 5:** dừng vòng while đầu tiên

Nhảy vào vòng while thứ 2

Khi mà stack chưa trống:

* top = 4 => h[top] = 4;



stack.pop => Lúc này: stack.peek = 2



subarea = h[top] \* (i – stack.peek – 1) = 4\*2 = 8

max\_area = subarea = 8

* top = 2 => h[top] = 4



stack.pop => Lúc này: stack.peek = 1



**subarea = h[top] \* (i – stack.peek – 1) = 4\*(5-1-1) = 12 (max)**



Sau đó máy chạy nốt đến khi stack rỗng



