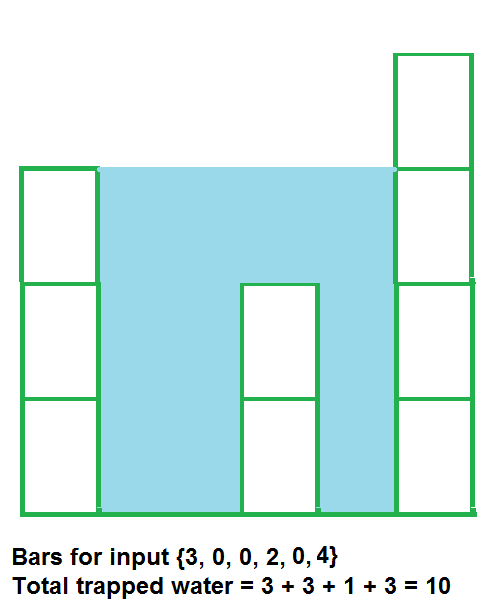
Given an array **arr[]** of **N** non-negative integers representing the height of blocks at index **i** as **Ai** where the width of each block is 1. Compute how much water can be trapped in between blocks after raining.  
**The structure is like below**:  
|  |  
|\_|  
We can trap 2 units of water in the middle gap.



**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows. Each test case contains an integer **N** denoting the size of the array, followed by **N**space-separated numbers to be stored in the array.

**Output:**  
Output the total unit of water trapped in between the blocks.

**User Task:**  
The task is to complete the function **trappingWater**() which returns the total amount of water that can be trapped.

**Expected Time Complexity:**O(N).  
**Expected Auxiliary Space:**O(N).

**Constraints:**  
1 <= T <= 100  
3 <= N <= 107  
0 <= Ai <= 108

**Example:  
Input:**  
2  
4  
7 4 0 9  
3  
6 9 9

**Output:**  
10  
0

**Explanation:  
Testcase 1:** Water trapped by the block of height 4 is 3 units, block of height 0 is 7 units. So, the total unit of water trapped is 10 units.

**Implementation using stack:**

**#include<bits/stdc++.h>**

**using namespace std;**

**int trappingWater(int arr[], int n){**

**int ans=0;**

**stack<int>cont;**

**int dist,topi,bound\_cond;**

**//dist is width of rect**

**//bound\_cond is height of each rect**

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

**{**

**while(!cont.empty()&&arr[i]>arr[cont.top()])**

**{**

**topi=cont.top();**

**cont.pop();**

**//if stack is empty then there is no other element to compare it with**

**//1 4 like this case it will not have any water**

**if(cont.empty())**

**break;**

**/\***

**\***

**\***

**\* \***

**\* \***

**cond like this**

**\*/**

**/\***

**\***

**\* \***

**\* \***

**\* \* \***

**0 1 2**

**\*/**

**dist=i-cont.top()-1;**

**bound\_cond=min(arr[i],arr[cont.top()])-arr[topi];**

**/\***

**\***

**\* 1 \***

**\* \* \***

**\* \* \***

**0 1 2**

**1st time= dist (2-0-1)\*(min(4,3)-2)==1 unit sq water**

**we will fill the water in one step as numbered above**

**\*/**

**ans+=bound\_cond\*dist;**

**}**

**//just in case if its empty or its smaller number**

**cont.push(i);**

**}**

**return ans;**

**}**

**int main(){**

**int t;**

**cin >> t;**

**while(t--){**

**int n;**

**cin >> n;**

**int a[n];**

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

**cin >> a[i];**

**}**

**cout << trappingWater(a, n) << endl;**

**}**

**return 0;**

**} //}**