*All codes must be commented and justified   
You get points if your solution is correct   
In order to have all the points, you must respect the complexity specifications*

1. Implement a data structure supporting the following operation:   
 - **bool** empty(): returns 1 if there is no element stored in the structure   
 - **void** add(**int** v): adds a new element equal to v in the structure   
 - **int** next(): returns and removes from the structure the upper median (i.e., if there are n = 2m elements stored, exactly m+1 elements should be lower than the output; if there are n=2m+1 elements, exactly m should be lower than the output).   
All the operations should run in O(log(n)) time **/5**

2. Consider the following implementation of a binary search tree:  
  
**typedef** **struct** BinaryTree {

**int** value;  
 **struct** BinaryTree \*tata;  
 **struct** BinaryTree \*stinga;  
 **struct** BinaryTree \*dreapta;  
} BinaryTree;  
  
 a) Write a function vector<**int**> productOfSubtree(BinaryTree \*T) which computes, for each node of T, the product of all the values stored in its subtree. The running time should be in O(n). **/3**

b) Write a function **int** productOfInterval(BinaryTree \*T, **int** x, **int** y, vector<**int**>& prod) which outputs the product of all the values stored in T between x and y. Here, the vector prod is the one computed at the previous question. The running time should be in O(height(T)). **/2**