*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. Consider the following simple data structure for a doubly linked list in C/C++:

**typedef struct** dlList {

**int** \_value;  
 **struct** dlList **\***pred;  
 **struct** dlList **\***urm;

} my\_list;

By using this code above, you are required to implement a data structure supporting the following operations:

* **bool** even(): returns 1 if the number of elements stored in the structure is even (possibly, 0) and 0 otherwise.
* **void** add(**int** v): adds a new element equal to v in the structure
* **int** next(): returns and removes, amongst the *first half* of all elements in the structure, the smallest one (an element e is in the first half of the structure if at least half of the elements in the structure were added after e).

Example: if we add 11,4,19,0,2,3 in the structure, and after that we call next(), then the element to be removed is 4.

All the operations should run in O(1) 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<**bool**> balancedSubtree(BinaryTree \*T) which verifies, for each node of T, whether its subtree is balanced. The running time should be in O(n). **/3**

b) Write a function **int** sumOfInterval(BinaryTree \*T, **int** x, **int** y, vector<**int**>& bal) which outputs the number of nodes whose subtree is balanced *and* whose value is between x and y.

Here, the vector bal is the one computed at the previous question. The running time should be in O(height(T)). **/2**