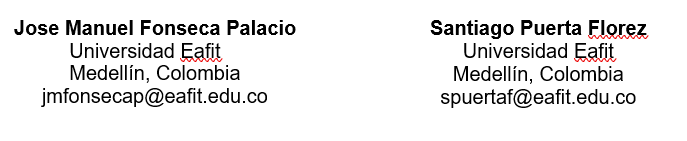
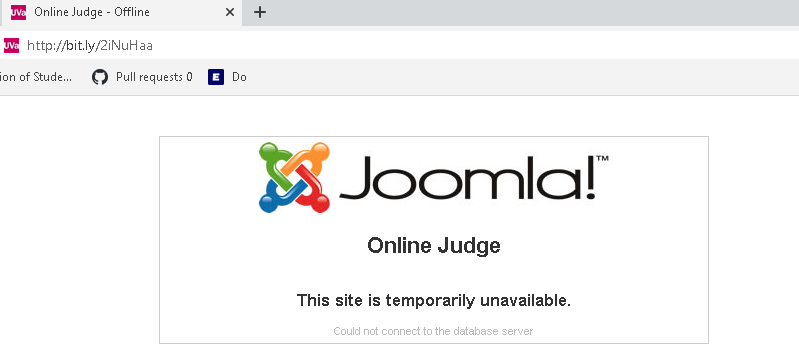
**Laboratory practice No. 4: Hash Tables and Trees.**



***2.2)*** ******

The problem couldn’t be solved due to the website is unavailable.

***3.1)*** We thought to manage the bees with a hash table, we chose this data structure because the time complexity was the priority and the hash table insertion time is O(1) and finding a required robotic bee is also done in O(1) time as well as the deletion of any robotic bee, so given the condition we thought that the hash table was the most adaptable choice.

***3.3) 2.1:*** The 2.1 works with n numbers that will be given by the user, the exercise tells us that those numbers are the pre-order path of the tree. This n numbers mentioned before are going to be taken as an array, so all that is left to do is fill up the tree with the array and print first the left node of the tree then the right node and finally the root. And that will give us the expected output.

2.2: Can’t be explained since the website was unavailable

***3.4) 2.1:*** The time complexity for the exercise 2.1 is O(log(n)) and the space complexity is O(n)

***2.2:*** 2.2’s complexity can’t be explained since the website was unavailable.

***3.5) 2.1:*** The “n” in the time complexity calculus stands for the right or left subset chosen by the condition: newVal < value (left) and value > newValue (right).

And the “n” in the space complexity calculus stands for the number of nodes that the tree is composed by.

***2.2:*** Can’t be explained since the website was unavailable

***4)***

***4.1a:*** b

***4.1b:*** d

***4.2:*** c

***4.3:***

***a) false;***

***b) a.data;***

***c)a.izq, suma-a.data***

***d)a.der, suma-a.data***

***4.4)***

***4.4.1:*** d

***4.4.2:*** a

***4.4.3:*** d

***4.4.4:*** a

***4.9:*** a

***4.12.1:*** i

***4.12.2:*** a

***4.12.3:*** b

***4.13)***

***4.13.1)raiz.id***

***4.13.2)a***