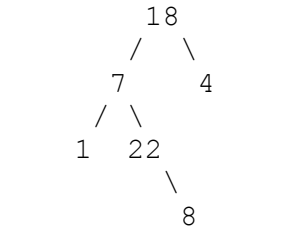
Name: _____
UCFID: _____
NID: _____

1) (10 pts) DSN (Binary Trees)

Write a function named *find_below*() that takes a pointer to the root of a binary tree (*root*) and an integer value (*val*) and returns the greatest value in the tree that is strictly less than *val*. If no such value exists, simply return *val* itself. Note that the tree passed to your function will **not** necessarily be a binary **search** tree; it's just a regular binary tree.

For example:




```
find_below(root, 196) would return 22
find_below(root, 1) would return 1
find_below(root, 4) would return 1
find_below(root, 22) would return 18
find_below(root, 20) would return 18
find_below(root, 8) would return 7
find_below(root, -23) would return -23
```

You must write your solution in a **single** function. You cannot write any helper functions.

The function signature and node struct are given below.

```
typedef struct node
{
   int data;
   struct node *left;
   struct node *right;
} node;

int find_below(node *root, int val)
{
      int retval = val;
      int v1, v2;

      // Grading 2 pts for NULL case.
      if (root == NULL)
            return val;

      // Grading: 2 pts each, to make both recursive calls and store answers.
      v1 = find_below(root->left, val);
      v2 = find_below(root->right, val);

      // 4 pts total for this logic. 2 pts when answer is val, 1 pt each for v1, v2.
      if (root->data < val && (root->data > retval || retval == val)) retval = root->data;
      if (v1 < val && (v1 > retval || retval == val)) retval = v1;
      if (v2 < val && (v2 > retval || retval == val)) retval = v2;

      return retval;
}
// Note: Many ways to do the logic at the end.
```