

1) (10 pts) DSN (Binary Search Trees)

Complete the function on the next page so that it takes the root of a binary search tree (*root*) and an integer (*key*) and, if *key* is in the tree, returns the smallest integer in the BST that is **strictly greater than** *key*. If *key* is not present, or if there is no integer in the tree greater than *key*, simply return *INT_MIN* (which is defined in *limits.h*).

Your function must be **iterative** (not recursive), with a worst-case runtime that does not exceed **O(n)** (so, you can't drop the elements into an array and sort them in order to solve the problem).

You may assume the tree does not contain any duplicate values. However, *root* could be NULL.

For example:

<pre> 18 / \ 4 20 / \ 1 10 / 7 </pre>	<pre> next_up(root, 18) would return 20 next_up(root, 1) would return 4 next_up(root, 4) would return 7 next_up(root, 10) would return 18 next_up(root, 20) would return INT_MIN next_up(root, 9) would return INT_MIN </pre>
---	---

In your solution, you may make as **single call** to either of the following functions. Assume they're already written and that they each have a worst-case runtime of O(n):

```

// Takes the root of a binary search tree (possibly the root of a
// subtree within a larger BST) and returns the smallest value in that
// (sub)tree. If the tree is empty, it returns INT_MAX.
int find_min(node *root);

// Takes the root of a binary search tree (possibly the root of a
// subtree within a larger BST) and returns the largest value in that
// (sub)tree. If the tree is empty, it returns INT_MIN.
int find_max(node *root);

```

An incomplete version of the function and *node* struct are provided on the following page, along with ten blanks for you to fill in to complete the solution. **Note that one of these blanks ought to be left blank and has been included so that part of the solution isn't given away.** Thus, each blank is worth one point, and for at least one of the ten blanks, leaving it blank is the only way to get credit for it.

(...continued from previous page)

```
// Assume these are included from limits.h.
#define INT_MAX 2147483647
#define INT_MIN -2147483648

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

int next_up(node *root, int key)
{
    node *parent = NULL ;           // Grading: 1 pt per slot, blank slot can
                                     // be either of the two slots for else if

    while (root != NULL)
    {
        if (key < root->data)
        {
            parent = root ;

            root = root->left ;

        }
        else if (key > root->data)
        {
            root = root->right ;

            _____ ;
        }
        else
        {
            if ( root->right != NULL )

                return find min(root->right) ;

            else if (parent != NULL)

                return parent->data ;
            else
                return INT_MIN ;
        }
    }

    return INT_MIN ;
}
```