2) (10 pts) DSN (Linked Lists)

We can store an integer in a linked list of nodes, where each node stores digit, in reverse order. For example, the integer 2163 would be stored in the linked list 3 → 6 → 1 → 2. Using the node struct shown below that is used to store numbers in this manner, write a **recursive** compareTo function that takes in pointers to two integer stored in this manner and returns a negative integer if the number in the list pointed to by num1 is less than the number in the list pointed to by num2, 0 if the two respective numbers are equal, or a positive integer if the number in the list pointed to by num1 is larger than the number in the list pointed to by num2. For example, compareTo(3 → 6 → 1 → 2, 4 → 6 → 1 → 2) should return a negative integer and compareTo(3 → 6 → 1 → 2, 9 → 9 → 9 → 1) should return a positive integer.

```
typedef struct node {
    int digit;
    struct node* next;
} node;


int compareTo(node* num1, node* num2) {

    if (num1 == NULL && num2 == NULL) return 0;

    if (num1 == NULL) return -1;

    if (num2 == NULL) return 1;

    int tmp = compareTo(num1->next, num2->next);

    if (tmp != 0) return tmp;

    return num1->digit - num2->digit;
}
```

Grading: 1 pt for each base case (order matters to avoid null ptr error)
3 pts for the recursive call
2 pts to return the recursive call answer when it's not 0.
2 pts to return appropriately when the least significant digit breaks the tie.

Note: The base cases can be done in a few ways.
Most students will probably use an if-else to compare num1->digit, num2->digt at the end there.