2) (10 pts) DSN (Linked Lists)

The structure of each node of a singly linked list is shown below.

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

Write a function insertAfterN, that takes the head of a linked list, and two integers M and N (M ≠ N) and inserts M after all the nodes containing N.

For example, if M = 200 and N = 6, the linked list 3, 6, 4, 6, 6, 5 will be changed to 3, 6, 200, 4, 6, 200, 6, 200, 5.

```
void insertAfterN(node* head, int M, int N) {

    if (head == NULL) return;

    if (head->data == N) {
        node* tmp = malloc(sizeof(node));
        tmp->data = M;
        tmp->next = head->next;
        head->next = tmp;
        head = tmp;
    }

    insertAfterN(head->next, M, N);

}
```

Grading:

4 pts for "looping" mechanism. (In this solution, that means both the base case, the recursive call, and making sure head->next is the next node. It's also okay if head=tmp; is not included.) An iterative solution would include a for or while loop with head advancing.

1 pt for checking if the data value in a node equals N

2 pts malloc new node in the appropriate case (1 pt malloc, 1 pt parameter and assignment)
1 pt store M in data field of new node
1 pt linking new node to the old next of the list
1 pt linking node storing N to node storing M