3) (10 pts) DSN (Stacks)

Suppose we have implemented a stack using a linked list. The structure of each node of the linked list is shown below. The stack structure contains a pointer to the head of a linked list and an integer, size, to indicate how many items are on the stack.

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

typedef struct Stack {
    struct node *top;
    int size;
} stack;
```

Write a function that will pop off the contents of the input stack and push them onto a newly created stack, returning a pointer to the newly created stack. In effect, your function should reverse the order of the items in the original stack, placing them in a new stack. Assume you have access to all of the usual stack functions. Assume that when you push an item onto the stack, its size automatically gets updated by the push function. Similarly for pop, size gets updated appropriately when you pop an item from a stack. Do NOT call pop or peek on an empty stack.

```
void push(stack *s, int number); // Pushes number onto stack.
int pop(stack *s);  // Pops value at top of stack, and returns it.
int peek(stack *s); // Returns value at top of stack.
int isEmpty(stack *s); // Returns 1 iff the stack is empty.


stack* reverseStack(stack* s) {

    stack *newS = malloc(sizeof(stack));

    news->size = 0;            // 2 pts
    newS->top = NULL;          // 2 pts

    while(!isEmpty(s))         // 3 pts
        push(newS, pop(s));    // 3 pts




    return newS;
}
```