Making a stack from a singly linked list (is easy)

**Push**
Add a new node at the front (head).

**Pop**
Remove the node currently at the front (head).
typedef struct node {
    int value;
    struct node *next;
} Node;

Needed for an empty stack:

Node *stackptr; /*head of List*/

stackptr = NULL;
**Step-by-Step for push:**

- New TOS
- Node *push (Node * oldTOS, int valueToPush)

1. Allocate new Node
   - Use `malloc()`
   - Check return value!

2. Set value field
3) hook node into list

3a) if list is empty

3b) if list is not empty

newnode

old TOS

existing list
4) return newnode;

Why not change oldTOS with
oldTOS = newnode;

?
For `pop()`, there are 2 issues to solve:

1. What if `pop()` is called with an empty stack?

2. We need to return 2 values.
   - The value field
   - new TOS (stack) with 1 fewer item

```c
int pop(Node **pstackptr)
```
INCOMPLETE pop() CODE

if (*pstackptr != NULL) {
    tempnode = *pstackptr;
    poppedint = tempnode->value;
    *pstackptr = tempnode->next;
    /* be a good citizen, and return */
    /* memory allocated by malloc() */
    free(tempnode);

    return poppedint;
}