Name: Login: IN-CLASS EXERCISE 3/4/2020

Linked lists

1. Let's make `append(…)`.

Fill in the blanks on lines 1, 3, 6, 9, 12, 14, 18, and 22. Answer is on the other side, but try not to look until you have made your best effort. You have already seen this code before. The only value here is in doing this yourself.

```
 1 void append(int value,                       ,                       ) {
 2     // Assert:  If head is NULL, then tail must be NULL, too.
 3     assert(                                      );
 4
 5     // Allocate space for a struct Node object using malloc(…).
 6     struct Node* new_node =                          ;
 7
 8     // Initialize the .value and .next fields.  Use a compound literal.
 9
10
11     // If list is empty...
12     if(               ) {  // If list is empty
13         // Set the head to the new node.
14
15     }
16     else {  // Not empty....
17         // Connect the .next field of the tail to the new node.
18
19     }
20
21     // Set the tail to the new node, in any case.
22
23 }
```

2. Draw after one node.

```
struct Node* head = NULL;
struct Node* tail = NULL;
append(3, &head, &tail);
```

3. Draw after four nodes.

```
struct Node* head = NULL;
struct Node* tail = NULL;
append(3, &head, &tail);
append(4, &head, &tail);
append(5, &head, &tail);
append(6, &head, &tail);
```

Answer to #1

```
 1 void append(int value, struct Node** a_head, struct Node** a_tail) {
 2     // Assert:  If head is NULL, then tail must be NULL, too.
 3     assert( (*a_tail == NULL) == (*a_head == NULL) );
 4
 5     // Allocate space for a struct Node object using malloc(…).
 6     struct Node* new_node = malloc(sizeof(*new_node));
 7
 8     // Initialize the .value and .next fields.  Use a compound literal.
 9     *new_node = (struct Node) { .value=value, .next=NULL };
10
11     // If list is empty...
12     if(*a_tail == NULL) {  // If list is empty
13         // Set the head to the new node.
14         *a_head = new_node;
15     }
16     else {  // Not empty....
17         // Connect the .next field of the tail to the new node.
18         (*a_tail) -> next = new_node;
19     }
20
21     // Set the tail to the new node, in any case.
22     *a_tail = new_node;
23 }
```