Linked Stack Push Operation

Assume that we have the following lines of code:

```c
mystack stack1; // Line 1

stack1.push(5); // Line 2
stack1.push(8); // Line 3
stack1.push(3); // Line 4
```

The following sequence of diagrams shows how the Stack object and its associated dynamic storage changes as these lines are executed.

**Figure 1:** The new, empty mystack object `stack1` created in Line 1 of the code above. The stk_top pointer is `nullptr`, while stk_size is 0.

```
stk_top   X
stk_size   0
```

**Figure 2a:** The call to `push()` in Line 2 causes a new list `Node` to be allocated using the temporary pointer `new_node`. The node's value field is initialized to the value passed to `push()`, while its next field is initialized to the current value of stk_top.

```
stk_top   X
stk_size   0
```

```
5     X
new_node
```

**Figure 2b:** The pointer stk_top is set to point at new_node and the stk_size is incremented to 1.

```
stk_top   X
stk_size   1
```

```
5     X
new_node
```
**Figure 2c:** When the `push()` method ends, the local variable `new_node` ceases to exist.

**Figure 3a:** The call to `push()` in Line 3 causes a new list `Node` to be allocated using the temporary pointer `new_node`. The node's `value` field is initialized to the value passed to `push()`, while its `next` field is initialized to the current value of `stk_top`.

**Figure 3b:** The pointer `stk_top` is set to point at `new_node` and the `stk_size` is incremented to 2.

**Figure 3c:** When the `push()` method ends, the local variable `new_node` ceases to exist.
Figure 4: Linked stack following the call to push() in Line 4.

```
stack1

stk_top
3

stk_size
3
```

Linked Stack Pop Operation

Assume that we then add the following lines of code after the code listed above:

```
stack1.pop();       // Line 5
stack1.pop();       // Line 6
stack1.pop();       // Line 7
```

The following sequence of diagrams shows how the mystack object and its associated dynamic storage changes as these lines are executed.

Figure 5a: The call to pop() in Line 5 creates the temporary pointer del_node and sets it to the value of stk_top.

```
stk_top
3

stk_size
3
```

```
stk_top

3

stk_size
3
```

Figure 5b: The pointer stk_top is set to stk_top->next. It now points to the 2nd node in the list.

```
stk_top
8

stk_size
3
```

```
stk_top

8

stk_size
3
```

```del_node
stk_top

3
```

```
stk_top

3
```

```
stk_top

8

5

X
```

```del_node
stk_top

8

5

X
```
**Figure 5c:** The node pointed to by del_node is deleted and stk_size is decremented to 2.

![Diagram showing deletion of node and updating stk_size]

**Figure 5d:** When the pop() method ends, the local variable del_node ceases to exist.

![Diagram showing updated stk_top and stk_size]

**Figure 6a:** The call to pop() in Line 6 creates the temporary pointer del_node and sets it to the value of stk_top.

![Diagram showing creation of del_node and setting to stk_top]

**Figure 6b:** The pointer stk_top is set to stk_top->next. It now points to the 2nd node in the list.

![Diagram showing updated stk_top and stk_size]
**Figure 6c:** The node pointed to by del_node is deleted and stk_size is decremented to 1.

**Figure 6d:** When the pop() method ends, the local variable del_node ceases to exist.

**Figure 7a:** The call to pop() in Line 7 creates the temporary pointer del_node and sets it to the value of stk_top.

**Figure 7b:** The pointer stk_top is set to stk_top->next. It is now nullptr.
Figure 7c: The node pointed to by del_node is deleted and stk_size is decremented to 1.

Figure 7d: When the pop() method ends, the local variable del_node ceases to exist. The stack is now empty.