LW: Linked List Implementation of Stack

Do This First:

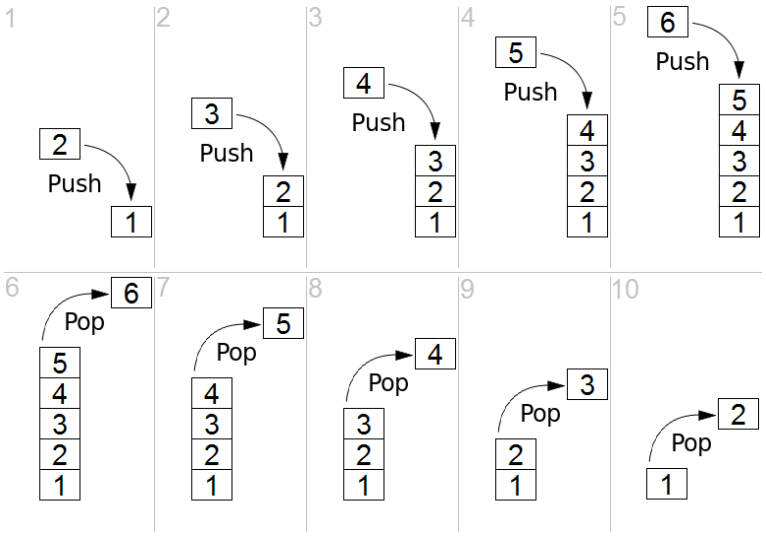
Download the starter code and submit it to Mimir. This will allow you to see the test cases.

In Lab Work 7 (Stack and Stack Applications), you were introduced to the *stack* abstract data type, which you implemented as a struct and functions which operated on a struct Stack. In Lab Work 11 (Stack with Class and Class Template), you implemented a stack of doubles as a class and then implemented a templated stack class. In both of those LWs, you implemented a stack using an array.

Recall from LWs 7 and 11 that a stack is a collection of elements which has two principal operations:

1. `push` : add an element at the top of the stack
2. `pop` : remove the element at the top of the stack

A stack may also have a `peek` operation, which gives access to the top element without removing it.



Simple representation of a stack runtime with push and pop operations.
(https://en.wikipedia.org/wiki/Stack_(abstract_data_type)#/media/File:Lifo_stack.png)

In this Lab Work, you must implement a stack using a Linked List.

Choose a difficulty level:
Level 1: LinkedList<T> provided in full. Starter Code
Level 2: LinkedList<int> partially provided (not a template, missing some methods). Starter Code
Level 3: Start from scratch. Starter Code

Task 0 (Levels 2 and 3): Implement LinkedList<T>

Level 2: Template and Rule of Three

You must convert the provided LinkedList class to a template and implement the missing rule of three methods for the templated LinkedList class:

- `LinkedList(const LinkedList<T>&)` - copy constructor
- `LinkedList<T>& operator=(const LinkedList<T>&)` - copy assignment
- `~LinkedList()` - destructor

Level 3: From Scratch!

You must implement a templated LinkedList class that has (at least) the following methods:

- Rule of Three
- `void push_front(const T&)`
- `void pop_front()`
- `T& front() const`
- `unsigned int length() const`

Task 1 (All Levels): Implement Stack<T>

You must implement a templated Stack class that uses the templated Linked List class from Task 1 and has (at least) the following methods:

- `void push(const T&)`
- `void pop()`
    - Attempting to `pop` from an empty stack should throw an exception. But, you can assume for this lab that the calling code will never call `pop` on an empty stack.
- `T& peek() const`
    - Attempting to `peek` an empty stack should throw an exception. But, you can assume for this lab that the calling code will never call `peek` on an empty stack.
- `bool empty() const`
- `friend std::ostream& operator<<(std::ostream&, const Stack&)`
    - Note: this should be defined inside the class definition (i.e. inside the class Stack {...};)
    - Format: `head-->...-->...-->tail`

Challenge Task (Level 3): Fun With Operator Overloading

Challenge yourself by implementing these methods:

- `Stack<T>& Stack<T>::operator<<(const T&)`
  - Chainable push operator
  - E.g. `stack << 1 << 2 << 3 << 4;`
- `Stack<T>& Stack<T>::operator>>(T&)`
  - Chainable pop operator
  - E.g. `stack >> x >> y >> z;`
- `Stack<T>& Stack<T>::operator+=(const Stack<T>&)`
  - Stack another stack on top of this stack (keeping the order of the stacks in tact)
  - E.g. [5,3,0,9] += [8,6,7] ⇒ [8,6,7,5,3,0,9]
    - Top of stack is the leftmost element

Submission

You must submit exactly 2 files:

1. `LinkedList.h` - Templated Linked List
2. `Stack.h` - Templated Stack (implemented using Linked List)

Do not submit `test121.h` or `test_Stack.cpp`.