**[CS250-Data-Structures](https://github.com/Rachels-Courses/CS250-Data-Structures)**/[Assignments](https://github.com/Rachels-Courses/CS250-Data-Structures/tree/2017-01-Spring/Assignments)/[Labs](https://github.com/Rachels-Courses/CS250-Data-Structures/tree/2017-01-Spring/Assignments/Labs)/**Lab 05 - Stacks and Queues**/

**Included files**

* main.cpp (DONE)
* DoublyLinkedList.hpp (DONE)
* Queue/Queue.hpp
* Stack/Stack.hpp

**Introduction**

In this lab, you will create a Stack and a Queue in a few short lines of code. A *DoublyLinkedList* has already been implemented for you, and the Queue and Stack will each *inherit* from that class.

Using the functions from DoublyLinkedList, implement Stack and Queue's functions:

* Push
* Pop
* Take

main() is already implemented so that the user can select a queue or a stack and, using Polymorphism, it will dynamically create a new queue or stack and run those functions.

**The implementation of each function will be one line each.**

To explicitly call a function from the child class, you can prefix the function name with the parent class and template marker:

DoublyLinkedList<T>::FunctionThingy( blorp );

**Queue**

Remember that a Queue is a FIRST IN, FIRST OUT structure.

You will need to answer the following:

* When pushing a new item into a queue, where does the new item go? (front? back?)
* When popping an item out of a queue, which item is removed? (front? back?)
* When taking an item from a queue (without removing), which item is accesses? (front? back?)

**Stack**

Remember that a Stack is a FIRST IN, LAST OUT (or LAST IN, FIRST OUT) structure.

You will need to answer the following:

* When pushing a new item into a stack, where does the new item go? (front? back?)
* When popping an item out of a stack, which item is removed? (front? back?)
* When taking an item from a stack (without removing), which item is accesses? (front? back?)

**Sample output**

**Queue:**

--------------------------------

1. Queue or 2. Stack, or anything else to quit.

>> 1

Create queue

Add A, B, C, D

Size: 4

Take: A

Current list:

A FIRST

B

C

D LAST

Pop 1

Size: 3

Take: B

Current list:

B FIRST

C

D LAST

**Stack:**

--------------------------------

1. Queue or 2. Stack, or anything else to quit.

>> 2

Create stack

Add A, B, C, D

Size: 4

Take: D

Current list:

A FIRST

B

C

D LAST

Pop 1

Size: 3

Take: C

Current list:

A FIRST

B

C LAST