

List Implementation #2: \_\_\_\_\_

```

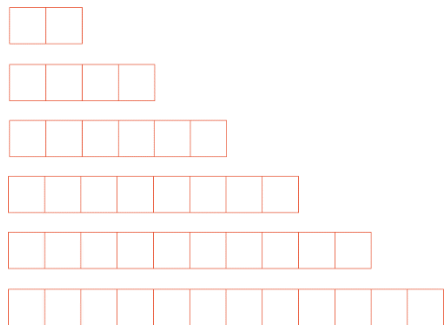
List.h
1 #pragma once
2
3 template <typename T>
4 class List {
5     public:
6     /* ... */
28     private:
29
30
31
32 };
    
```

Implementation Details and Analysis:

What is the running time of `insertFront()`?

C	S	2	2	5
[0]	[1]	[2]	[3]	[4]

→ What is our resize strategy?

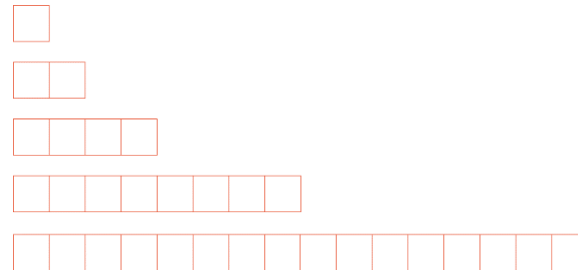


...total copies across all resizes: \_\_\_\_\_

...total number of insert operations: \_\_\_\_\_

...average (amortized) cost of copies per insert: \_\_\_\_\_

Array Resize Strategy #2:



...total copies across all resizes: \_\_\_\_\_

...total number of insert operations: \_\_\_\_\_

...average (amortized) cost of copies per insert: \_\_\_\_\_

Running Time:

	Singly Linked List	Array
Insert/Remove at <b>front</b>		
Insert after a <b>given</b> element		
Remove after a <b>given</b> element		
Insert at <b>arbitrary</b> location		
Remove at <b>arbitrary</b> location		

A List implementation in `std`

- `std::vector` implements a list with dynamic growth
- `#include <vector>` to use it!
- Documentation widely available, including on CBTF exams

## Stack ADT

Function Name	Purpose

## Queue ADT

Function Name	Purpose

## Stack and Queue Implementations

Stack.h	
1	#pragma once
2	
3	#include <vector>
4	
5	template <typename T>
6	class Stack {
7	public:
8	void push(T & t);
9	T & pop();
10	bool isEmpty();
11	
12	private:
13	std::vector<T> list_;
14	};
15	
16	#include "Stack.hpp"

Stack.hpp	
3	template <typename T>
4	void Stack<T>::push(const T & t) {
5	list_.push_back(t);
6	}
7	
8	template <typename T>
9	const T & Stack<T>::pop() {
10	const T & data = list_.back();
11	list_.pop_back();
12	return data;
13	}

## Three designs for data storage in data structures:

1. T & data
2. T \* data
3. T data

## Implication of Design

	Storage by Reference	Storage by Pointer	Storage by Value
Lifecycle management of data?			
Possible to insert NULL?			
External data manipulation?			
Speed			

## CS 225 – Things To Be Doing:

1. Programming Exam A starts Sept. 27 (*next Thursday*)
2. MP2 due Sept. 24 (*next Monday*); MP3 released Tuesday
3. lab\_inheritance due Sunday, reflections by Mattox on Monday
4. Daily POTDs