List Implementation #2: __________________________

```
List.hpp
103 template <typename T>
104 T List<T>::remove(unsigned index) {
105 
106 }
```

Implementation Details and Analysis:
What is the running time of `insertFront()`?

```
[0] [1] [2] [3] [4]
```

⇒ What is our resize strategy?

Array Resize Strategy #1:

...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:

<table>
<thead>
<tr>
<th>Insert/Remove at</th>
<th>Singly Linked List</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>front</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert after a <strong>given</strong> element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove after a <strong>given</strong> element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert at <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove at <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Queue ADT

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stack and Queue Implementations

```
#pragma once
#include <vector>
template <typename T>
class Stack {
public:
    void push(const T & d);
    T pop();
    bool isEmpty();
private:
    std::vector<T> list_;}
#include "Stack.hpp"
```

```
template <typename T>
void Stack<T>::push(const T & d) {
    list_.push_back(d);
}
template <typename T>
T Stack<T>::pop() {
    T data = list_.back();
    list_.pop_back();
    return data;
}
```

CS 225 – Things To Be Doing:

1. mp_stickers due Monday;
2. Daily POTDs