List Implementation #2: __________________________

```
#pragma once

template <typename T>
class List {
public:
    /* ... */
private:
};
```

Implementation Details and Analysis:

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

What is our resize strategy?

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

### Stack and Queue Implementations

**Stack.h**

```cpp
#pragma once

#include <vector>

template <typename T>
class Stack {
public:
    void push(T & t);
    T & pop();
    bool isEmpty();
private:
    std::vector<T> list_
};
```

**Stack.hpp**

```cpp
template <typename T>
void Stack<T>::push(const T & t) {
    list_.push_back(t);
}

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

### Three designs for data storage in data structures:

1. T & data

2. T * data

3. T data

### Implication of Design

<table>
<thead>
<tr>
<th>Lifecycle management of data?</th>
<th>Storage by Reference</th>
<th>Storage by Pointer</th>
<th>Storage by Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible to insert NULL?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External data manipulation?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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