A Linked List implementation of a List:

```cpp
#pragma once
template <typename T>
class List {
public:
    /* ... */
private:
    class ListNode {
    public:
        const T data;
        ListNode * next;
        ListNode(T & data) :
            data(data), next(nullptr) { }
    }
    ListNode *head_;
    /* ... */
};
```
What is our resize strategy?

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

<table>
<thead>
<tr>
<th>C</th>
<th>5</th>
<th>2</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0]</td>
<td>[1]</td>
<td>[2]</td>
<td>[3]</td>
<td>[4]</td>
</tr>
</tbody>
</table>

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>Function Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Insert/Remove at front</td>
<td></td>
</tr>
<tr>
<td>Insert after a <em>given</em> element</td>
<td></td>
</tr>
<tr>
<td>Remove after a <em>given</em> element</td>
<td></td>
</tr>
<tr>
<td>Insert at <em>arbitrary</em> location</td>
<td></td>
</tr>
<tr>
<td>Remove at <em>arbitrary</em> location</td>
<td></td>
</tr>
</tbody>
</table>

---

**Stack ADT**

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

---

**Queue ADT**

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

---

**CS 225 – Things To Be Doing:**

1. lab_memory starts today in person
2. mp_list released later today
3. Daily POTDs