List ADT

An abstract data type is a description of data being stored as well as valid operations that together describe a fundamental data type. A key part of this description is that implementation details are irrelevant.

What operations and data are required for the List ADT?

Two Basic Implementations of List:

1.

2.

Linked Memory:

| C | 5 | 2 | 2 | 5 | 5 | Ø |

Coding with Linked Lists: Examples

List.h

```cpp
#pragma once

#include "List.h"

template <typename T>
void List<T>::insertAtFront(T & t) {
    // Implementation...
}
```

List.hpp

```cpp
#include "List.h"

template <typename T>
typename List<T>::ListNode * List<T>::_index(unsigned index) {
    // Implementation...
}
```
Can you parse the meaning and logic behind the method:

typename List<T>::ListNode * & List<T>::_index(unsigned index)

Why did we choose to return type ‘T &’ for the [ ] operation?

How would you define a find() method using the same templated class?