CS 225

#3: List ADT

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:

List.h

```
28  class ListNode {
29    T & data;
30    ListNode * next;
31    ListNode(T & data) : data(data), next(NULL) { }
32  };
```

Coding with Linked Lists: Examples

List.h

```
 1  #pragma once
 2
 3  template <typename T>
 4  class List {
 5    public:
 6       /* ... */
 …    private:
28      class ListNode {
29        T & data;
30        ListNode * next;
31        ListNode(T & data) : data(data), next(NULL) { }
32      };
33
34
35
36
37
38
39  };
```

List.hpp

```
 9  #include "List.h"
10
11  template <typename T>
12  void List<T>::insertAtFront(T & t) {
13
14
15
16
17
18
19
20  }
```

```
39  template <typename T>
40  typename List<T>::ListNode *&
    List<T>::_index(unsigned index) {
41
42
43
44
45
46
47
48
49
50
51
52  }
```

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?