

### Two Basic Implementations of List:

- 1.
- 2.

### A Linked List implementation of a List:

```

List.cpp
1 #pragma once
2
3 template <typename T>
4 class List {
5     public:
6         /* ... */
7
8
9
10
11
12
13
14
15
16
17
18
19
20     private:
21         class ListNode {
22             public:
23                 const T data;
24                 ListNode * next;
25                 ListNode(T & data) :
26                     data(data), next(nullptr) { }
27
28         };
29
30     ListNode *head_;
31     /* ... */
32 };

```

### Implementing a basic List operation:



### Implementing a basic List operation:

```

List.hpp
9 #include "List.h"
10
11 ...
12
13
14 template <typename T>
15 void List<T>::insertAtFront(const T & d) {
16
17
18
19
20
21
22 }

```

### Finding in a list:



```

List.hpp
57 template <typename T>
58 typename List<T>::ListNode *&
59     List<T>::_index(unsigned index) {
60
61 }
62
63
64
65

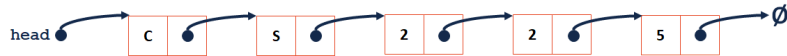
```

What is the return type of `_index`?

## Building functionality with `_index()`:

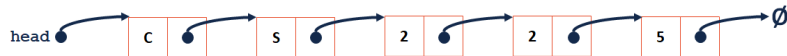
```

List.hpp
48 template <typename T>
49 T & List<T>::operator[](unsigned index) {
50
51
52
53
54 }
    
```



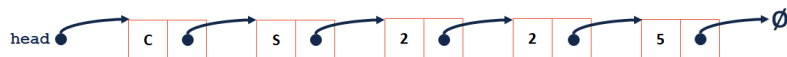
```

List.hpp
90 template <typename T>
91 void List<T>::insert(const T & t, unsigned index) {
92
93
94
95
96 }
    
```



```

List.hpp
103 template <typename T>
104 T List<T>::remove(unsigned index) {
105
106
107
108
109 }
    
```



## List Implementation #2: \_\_\_\_\_

```

Alternate List.h
1 #pragma once
2
3 template <typename T>
4 class List {
5 public:
6     /* ... */
7
28 private:
29
30
31
32 };
    
```

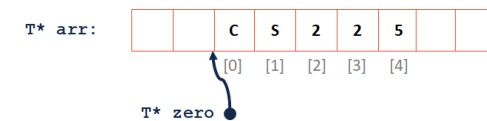
## Array - Implementation Details:



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



2. What is the running time of `get()`?



## CS 225 – Things To Be Doing:

1. No lab this week.
2. mp\_stickers EC **due tonight**;
3. Daily POTDs