Data Structures

Linked Lists

CS 225

Brad Solomon & G Carl Evans

August 28, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Discord Question Helpers

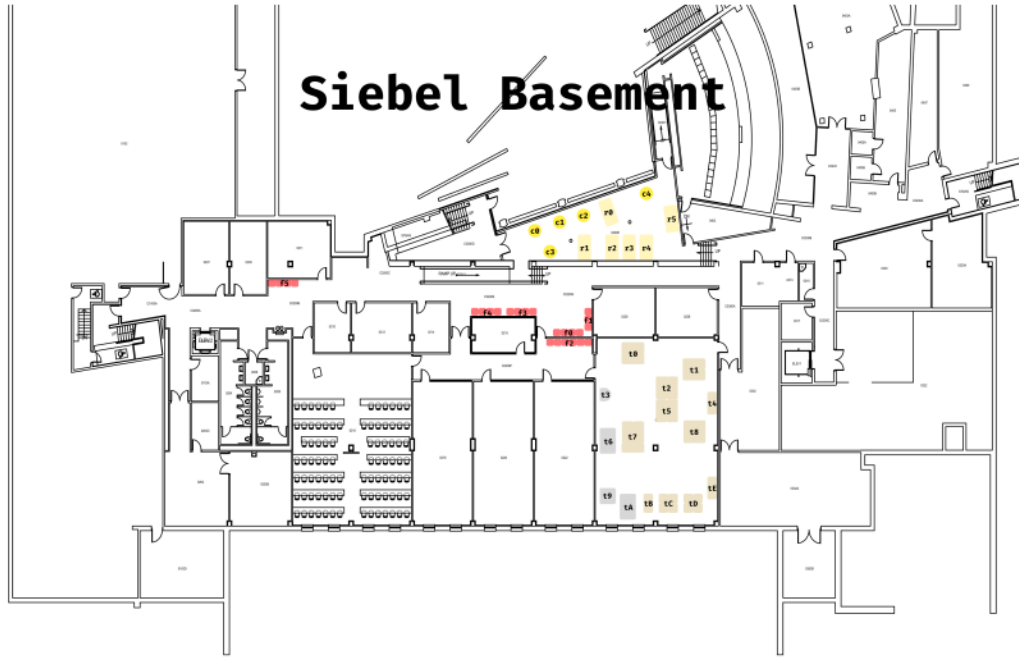
Glad to see so many people using Discord in lecture

To help answer questions in class, we will have staff members monitoring Discord.

Office Hour Etiquette

Schedule and link to queue on the website

Pay attention to the rules!



1. Be in Siebel Basement
2. Tag questions
3. Ask **one** specific question
4. Include a specific location
5. Include both your name and Discord ID

Learning Objectives

Review linked list operations (and go over new ones)

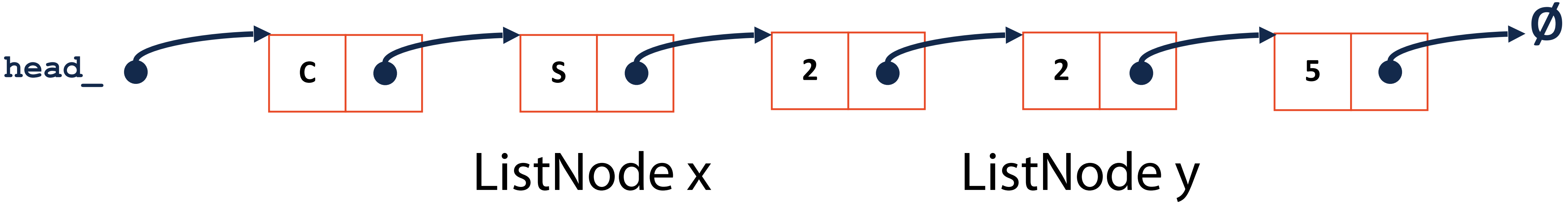
Introduce array list implementations

List.h

```
template <class T>
class List {
  public:
    /* ... */
  private:
    class ListNode {
      T & data;
      ListNode * next;
      ListNode(T & data) :
       data(data), next(NULL) { }
    };

    ListNode *head_;
};
```

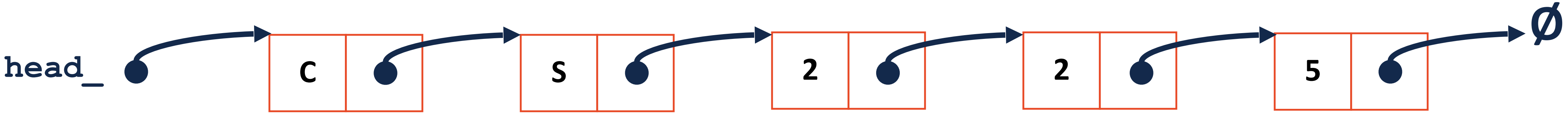
Can we access **x** from **y**?

Can we access **y** from **x**?



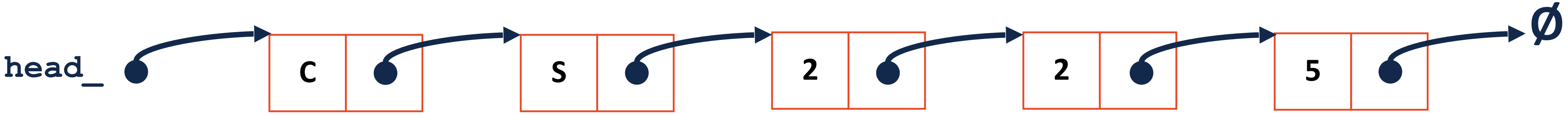
Linked List: `_index(index)`

`_index(0)`



Linked List: `_index(index)`

_index(2)



List.hpp

```
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index){
      return _index(index, head_)
}
```

```
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index, ListNode *& root){
















}
```

List.hpp

```
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index){
        return _index(index, head_)
}
```

```
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index, ListNode *& root){



    if (index == 0){ return root; }



    if (root == nullptr){ return root; }



    return _index(index - 1, root -> next);



}
```

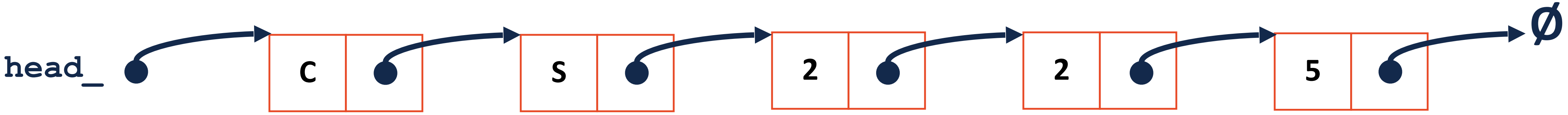
List.hpp

```cpp
// Iterative Solution:
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index) {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
       thru = thru->next;
    }
    return thru->next;
  }
}
```

What is the running time for iterative index?

What is the running time for recursive index?

Linked List: `insert(data, index)`



List.hpp

```
template <typename T>
void List<T>::insertAtFront(const T& t)
{
  ListNode *tmp = new ListNode(data);

  tmp->next = head_;

  head_ = tmp;

}
```

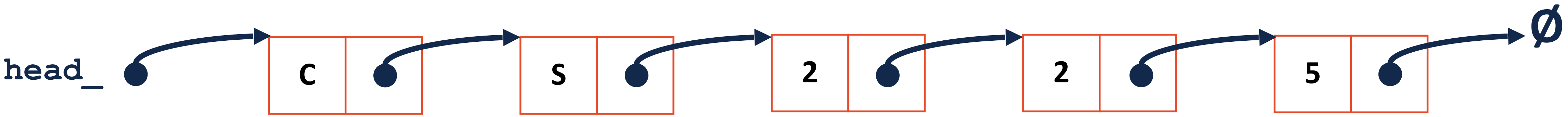
```
template <typename T>
void List<T>::insert(const T & data,
unsigned index) {



  ListNode *& curr = _index(index);



  ListNode * tmp = new ListNode(data);



  tmp->next = curr;



  curr = tmp;
}
```

List Random Access [ ]

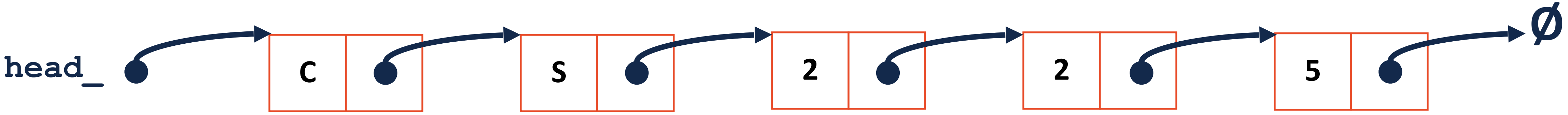
Given a list L, what operations can we do on L[ ]?

List.hpp

```
template <typename T>
T & List<T>::operator[](unsigned index) {









}
```

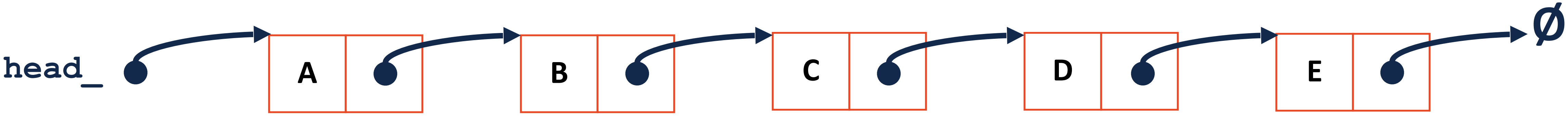


Linked List: `find(data)`

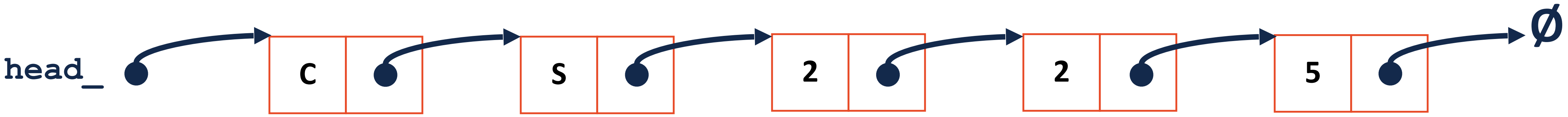


Linked List: Remove(<parameters>)

What input parameters make sense for remove?



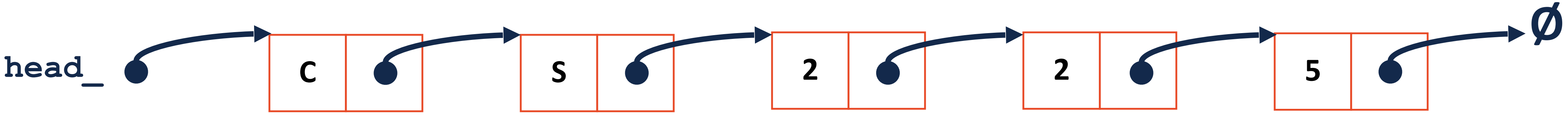
Linked List: `remove(data)`



List.hpp

```
template <typename T>
T List<T>::remove(ListNode *& node) {







}
```

Linked List: `remove`

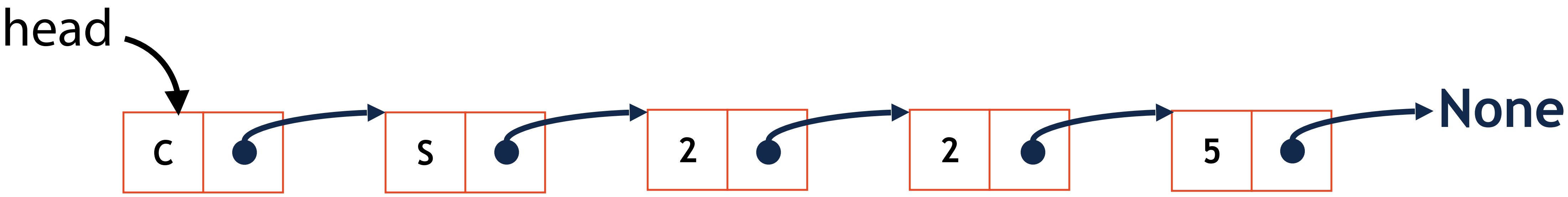


What is the running time to remove (if given a reference to a pointer)?

What is the running time to remove (if given a value)?

List Implementations

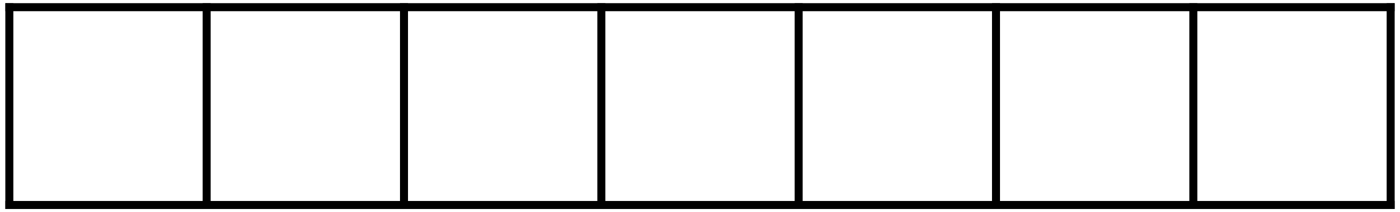
1. Linked List



2. Array List

C	S	2	2	5					

Array List



List.h

```
#pragma once

template <typename T>
class List {
public:
    /* --- */
private:
  T *data_;

  T *size;

  T *capacity;

    /* --- */
};
```