# CS 225

**Data Structures**

*January 19 – C++ and Lists in CS 225*

*G Carl Evans*

# Exam 0 Starts Monday

Covers prerequisite knowledge details on course website

https://courses.engr.illinois.edu/cs225/sp2023/exams/exam0/

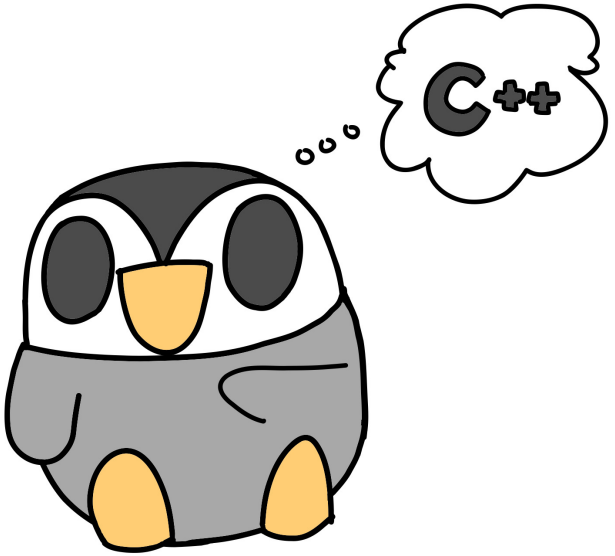
Practice on PrairieLearn

Register on PrairieTest

# Contacting the Course

cs225admin@lists.cs.illinois.edu

# What about C++




Lectures from Previous Semesters Covering C++ Available Here
https://mediaspace.illinois.edu/playlist/dedicated/177553201/1_s10ctiib/1_z2cz05fi
Material from CS 128
https://learncpp.online/lessons

# Encapsulation - Classes

# Memory Management - Ownership

# The “Rule of Three/Five”

If it is necessary to define any one of these three functions in a class, it will be necessary to define all three of these functions:

1.

2.

3.

# The “Rule of Zero”

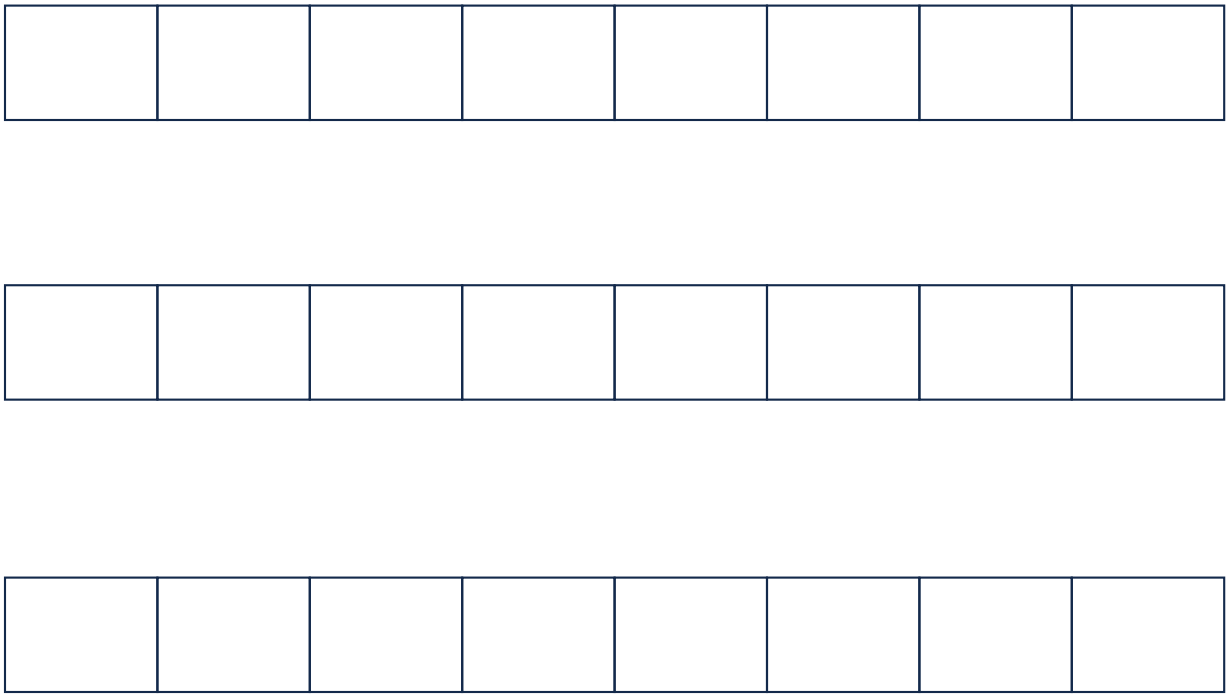
## Corollary to Rule of Five

Classes that **declare** custom destructors, copy/move constructors or copy/move assignment operators should deal exclusively with ownership. Other classes should not **declare** custom destructors, copy/move constructors or copy/move assignment operators

–Scott Meyers

# List ADT

# What types of “stuff” do we want in our list?



# Templates

## template1.cpp

```
1
2
3  T maximum(T a, T b) {
4    T result;
5    result = (a > b) ? a : b;
6    return result;
7  }
```

# List Implementations

1.

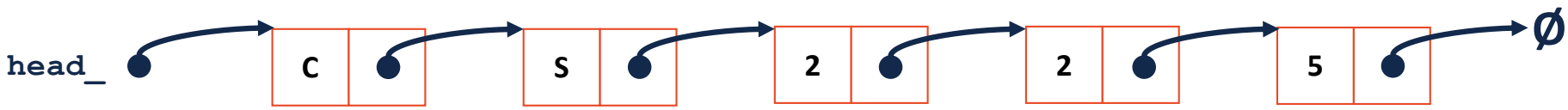
2.

# Linked Memory

## List.h

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

# Linked Memory



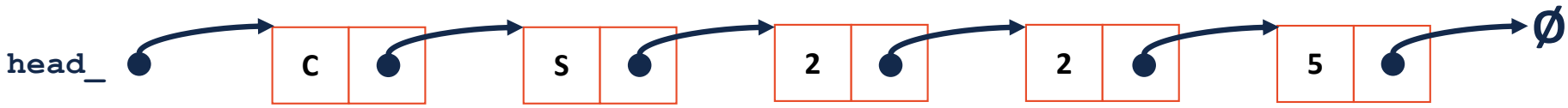
List.h

```
1  #pragma once
2
3
4  class List {
5    public:
6
7
8
9
10
11
12
13
14   private:
15
16
17
18 };
19
20
21
22
```

List.hpp

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```

# Linked Memory : `insertAtFront`



## List.h

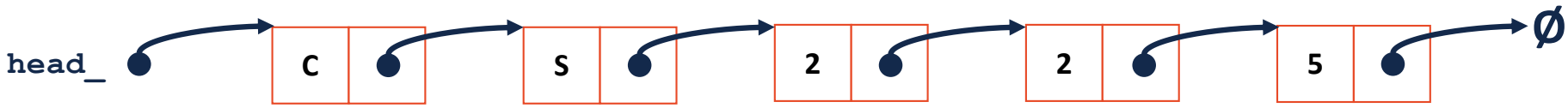
```
 1  #pragma once
 2
 3  template <class T>
 4  class List {
 5    public:
…      /* ... */
28    private:
29      struct ListNode {
30        T data;
31        ListNode * next;
32        ListNode(const T & data) :
           data(data), next(NULL) { }
33      };
34
35
36
37
38
39  };
40
41
```

## List.hpp

```
 1  #include "List.h"
 2
 3  template <class T>
 4  void List<T>::insertAtFront(const T& t) {
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22  }
```

# Running Time of Linked List `insertAtFront`

# Linked Memory : `insertAtNth`



List.hpp

```
33 ListNode *& List<T>::_index(int index) {
34
35
36
37
38
39
40 }
```