CS 225

Data Structures

Feb. 7 – List Implementation

Wade Fagen-Ulmschneider

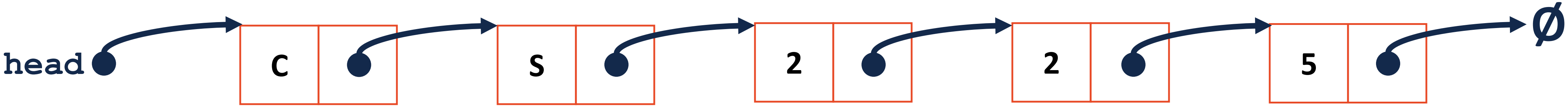
List.h

```
#ifndef LIST_H_
#define LIST_H_

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

    ListNode *head_;


};

#endif
```

List.cpp

```
#include "List.h"

template <typename T>
void List::insertAtFront(const T& t) {
  ListNode *e = new ListNode(t);
  e->next = head_;
  head_ = e;














}
```

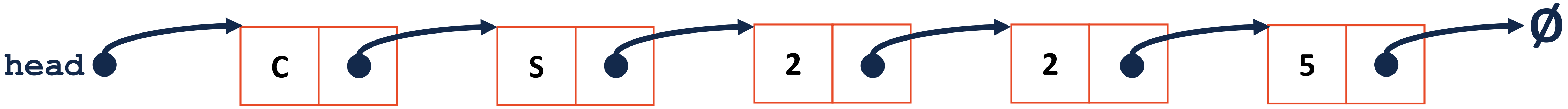
Linked Memory



List.cpp

```
#include "List.h"

ListNode *& List::_find(unsigned index) const {






}
```

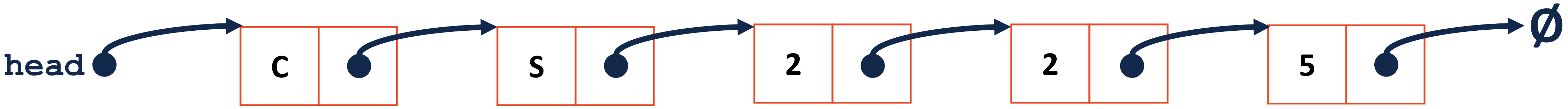
Linked Memory



List.cpp

```cpp
#include "List.h"

ListNode *& List::_find(unsigned index) const {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
       thru = thru->next;
    }
    return thru->next;
  }
}
```

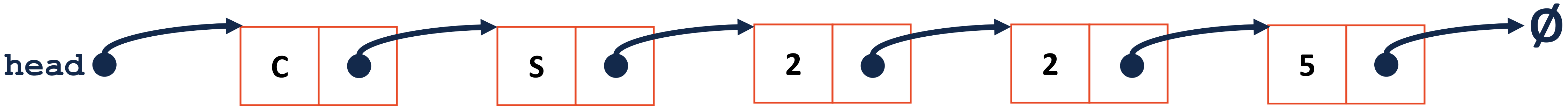
Linked Memory



List.cpp

```cpp
#include "List.h"

ListNode *& List::_find(unsigned index) const {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
        thru = thru->next;
    }
    return thru->next;
  }
}

template <typename T>
T & List::get(unsigned index) const {






}
```

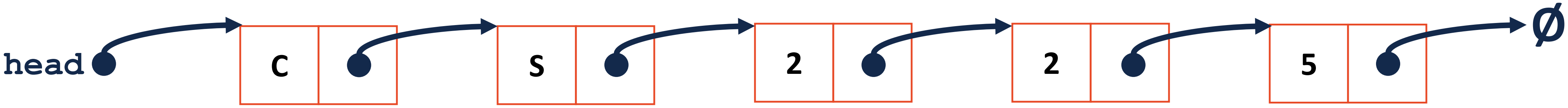
Linked Memory



List.cpp

```cpp
#include "List.h"

ListNode *& List::_find(unsigned index) const {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
        thru = thru->next;
    }
    return thru->next;
  }
}

template <typename T>
T & List::insert(T & t, unsigned index) {






}
```

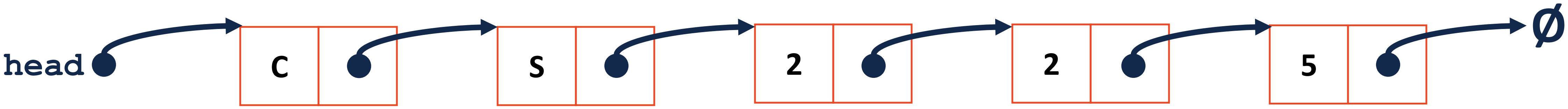
Linked Memory



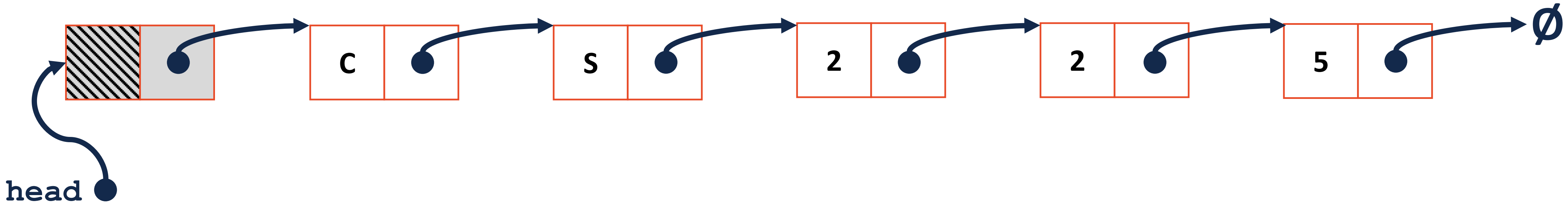
List.cpp

```
#include "List.h"

ListNode *& List::_find(unsigned index) const {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
       thru = thru->next;
    }
    return thru->next;
  }
}

template <typename T>
T & List::remove(unsigned index) {






}
```

Linked Memory



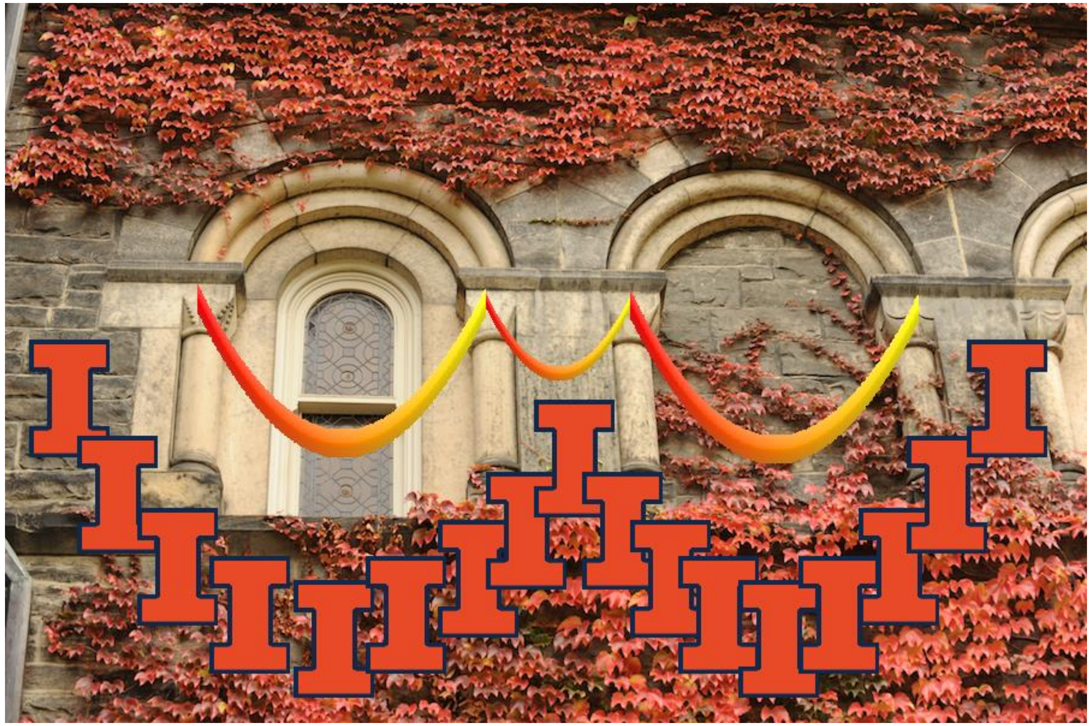
Sentinel Node



MP2

MP2

MP2



LEFT
EXIT 12
Making memes the normal way
Making memes using code written for CS225 hw
Welcome to CS225, this is Data Structures
www.drawyourprofessor.com
ILLINOIS
CS
225

MP2

MP2

List Implementations

1. **Linked List**

2.

List.h

```
#ifndef LIST_H
#define LIST_H

template <typename T>
class List {
  public:
    /* ... */
  private:












};

#endif
```

Array Implementation

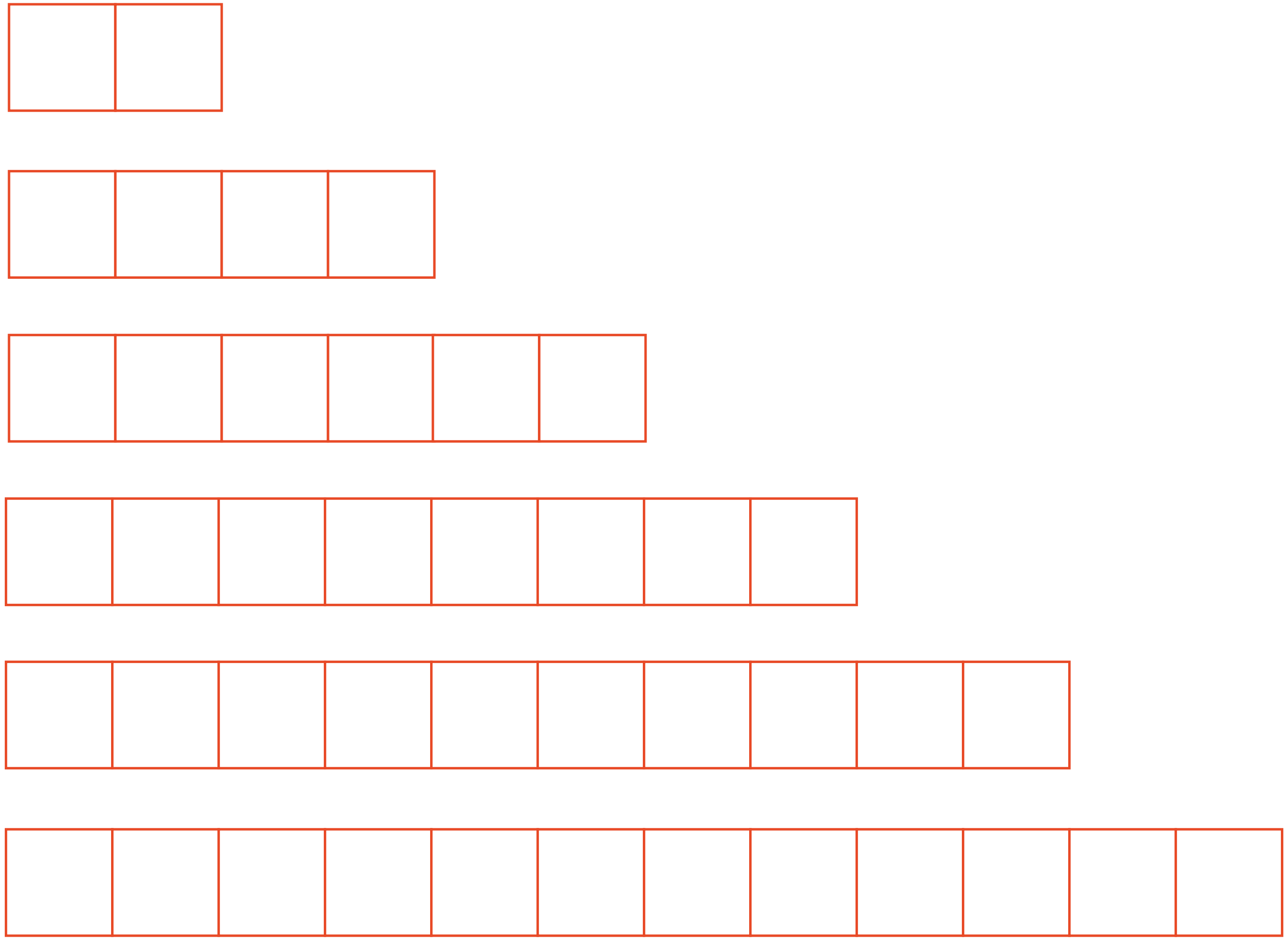
C	**S**	**2**	**2**	**5**
[0]	[1]	[2]	[3]	[4]

Array Implementation

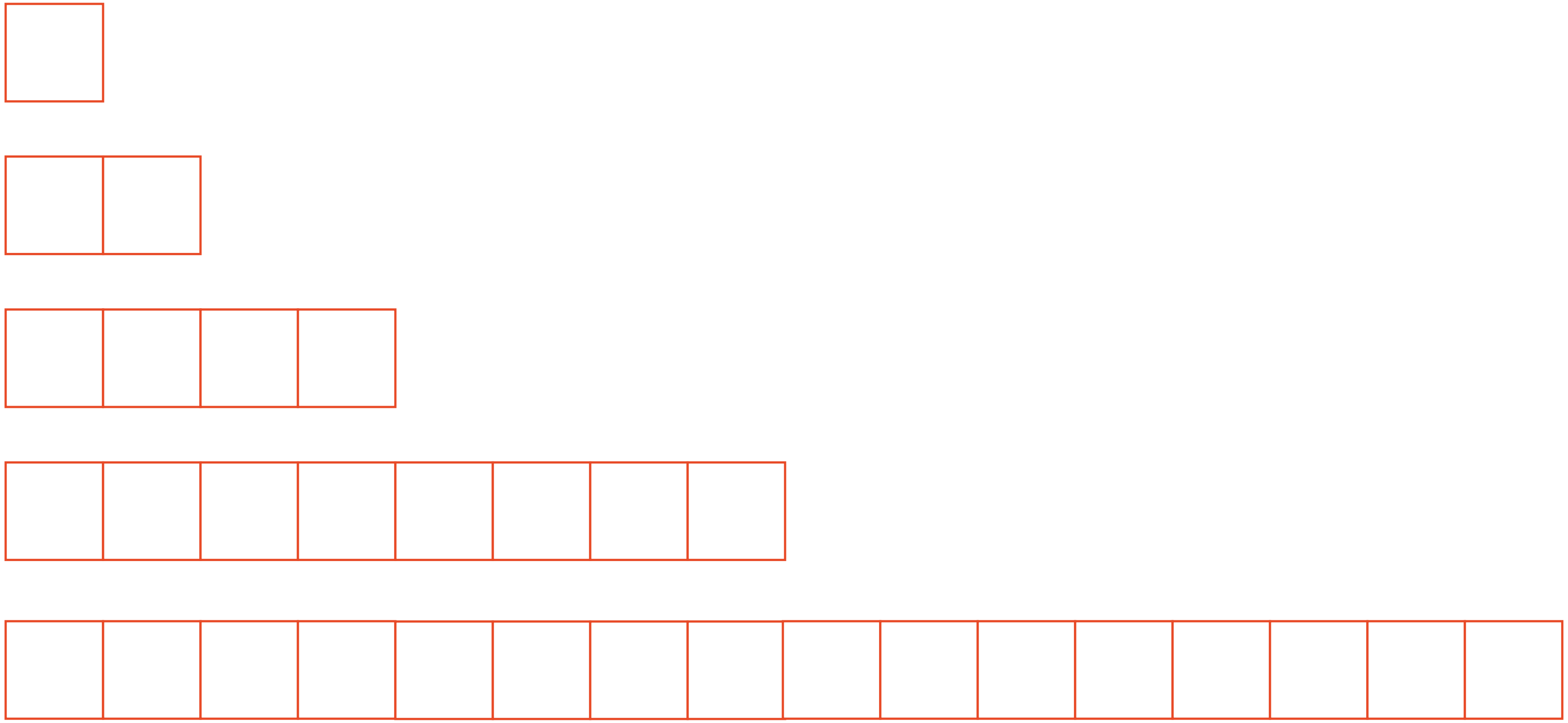
insertAtFront:

C	S	2	2	5
[0]	[1]	[2]	[3]	[4]

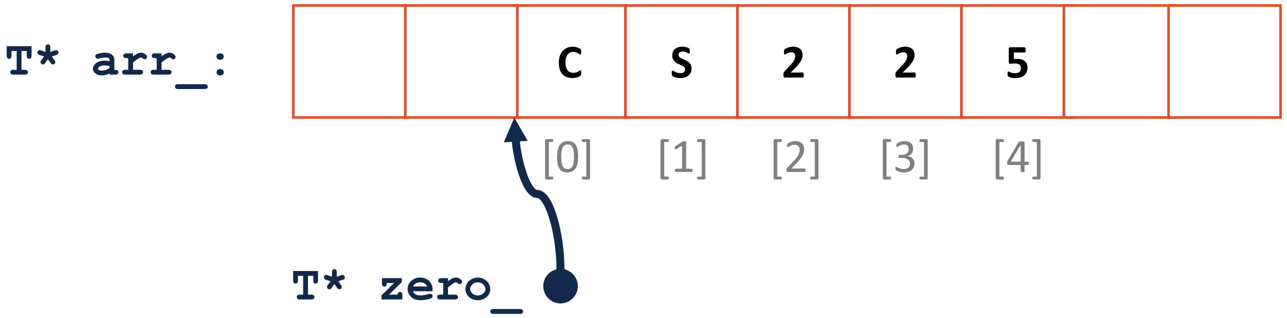
Resize Strategy – Details



Resize Strategy – Details



Array Implementation

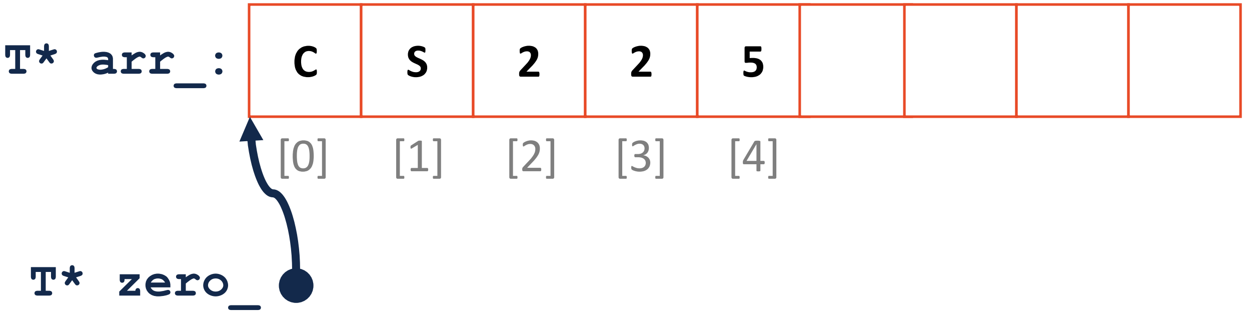


T* arr_:

		C	S	2	2	5		
		[0]	[1]	[2]	[3]	[4]		

T* zero_

Array Implementation

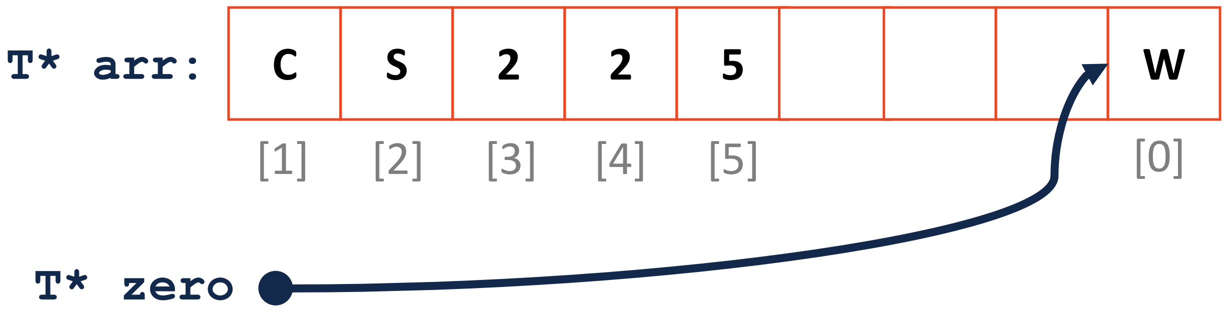


T* arr_:

C	S	2	2	5				
[0]	[1]	[2]	[3]	[4]				

T* zero_

Array Implementation

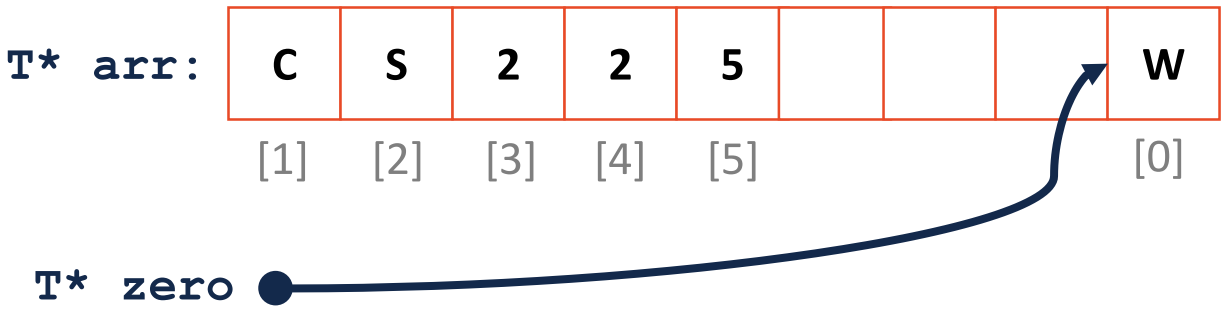


T* arr:	C	S	2	2	5				W
	[1]	[2]	[3]	[4]	[5]				[0]

T* zero

```
ListNode *& List::_get(unsigned index) const {
  return arr_[ (zero_ - arr_) + index % capacity_ ];
}
```

Array Implementation



T* arr:

C S 2 2 5 W

[1] [2] [3] [4] [5] [0]

T* zero

```
ListNode *& List::_get(unsigned index) const {
  return arr_[ (zero_ - arr_) + index % capacity_ ];
}
```

Array Implementation

	Singly Linked List	Array
Insert/Remove at **front**		
Insert at **given** element		
Remove at **given** element		
Insert at **arbitrary** location		
Remove at **arbitrary** location		