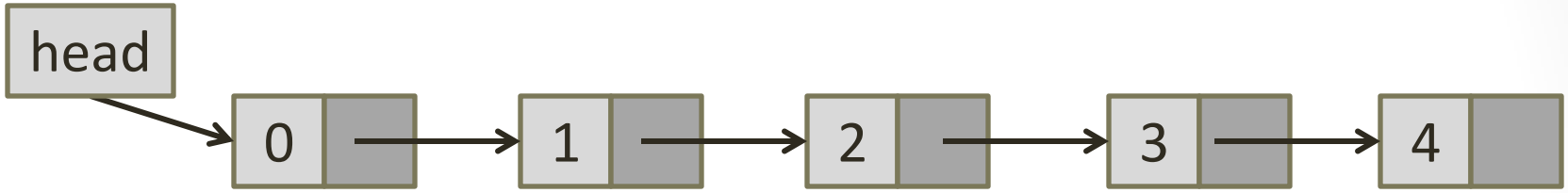
ECE220
Honors Lab Section

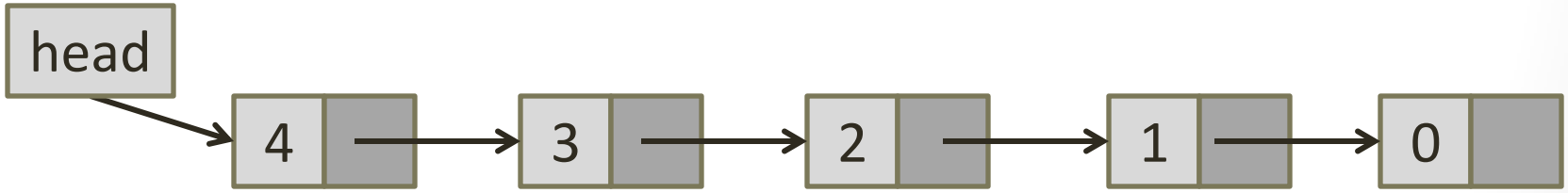
Lab 9: Linked list and generic data structures

Reversal

- Original linked list



- Reversed linked list



- How?

Reversal

- reverse_list(Node *head)
  - Node *rev = NULL, *cur = head, prev = NULL
  - while (cur != NULL)
    - prev = cur
    - cur = cur->next
    - rev.insert_at_front(prev)
  - return rev

- Runtime, memory space?
  - O(n), O(1) [we don't allocate any new Nodes and simply resuse]

Insertion

- Inserting nodes at front is easy. What about anywhere else?

- insert_single_pointer(Node *head, Node *new, int pos)
  - int idx = 0;
  - Node *cur = head, *prev = NULL;
  - while (cur !=NULL && idx++ < pos)
    - prev = cur
    - cur = cur->next
  - if (prev == NULL)
    - new->next = head
    - return new
  - else
    - prev->next = new
    - new->next = cur

Insertion

- Can we simplify this code?

- insert_double_ptr(Node *head, Node *new, int pos)
  - int idx = 0
  - Node **cur = &head
  - while (cur != NULL && idx++ < pos)
    - cur = &((*cur)->next)
  - new->next = *cur
  - *cur = new

- insert_single_ptr is 11lines
- insert_double_ptr is 6 line and handles edge cases easily

Function pointers

- Similar to regular pointers but for functions:
  - int (*add)(int x, int y)
    - defines add to be a function pointer with inputs x and y and return int
  - Node *(*allocate)(int data)
    - defines allocate to be a function pointer with input data and return a Node *
- Example:
  - int my_add(int a, int b)
    - return a + b
  - add = my_add
  - add(4, 5)

Function Pointers

- Function pointers in the real-world
  - void qsort(void *base, size_t nitems, size_t size, int (*compar)(const void *, const void*))

- The qsort() function is C implementation of quicksort.
  - Base: pointer to the first element
  - nitems: number of elements pointed to by base
  - size: size of each element in the array
  - (*compar): function pointer to function that compares elements in the base array

Generics

- How does one go about implementing a generic data structure like a queue for chars, ints, doubles?
- Void pointers!

- Generic queue demo:
  - Note that code can be found in honors_lab9 in the queue/stack.c/h files

Generics

- Review:
  - Create a data structure with some sort of void pointers
    - In the case of the queue/stack, I used an array so I had an array of void pointers.
  - Whenever you insert an element, pass in a function pointer that specifies how to allocate new element
  - Pass function pointers for other operations as well

- Look at the qsort() function from Slide 7. Uses void *'s as well to do sorting.

Generics

- Final thoughts
  - This is sort of reinventing the wheel and implementing classes in C (which is what C++ is all about).
  - You can create generic data structure much more easily in C++ with templates.