Day 18 (Fri 03/04)

- exercise 17 review
- day 18 recap questions
- exercise 18

Announcements/reminders

- HW4 due this evening by 11pm
  - written assignment, no late submissions

- midterm project:
  - you should be in a group
  - your group should have access to your team repository
  - contact me ASAP if either of above is not true

- midterm exam:
  - in class Friday 3/11
  - computer based
    - suggest using lab PC: there will be no accommodations if your personal laptop doesn't work (battery issues, network issues, etc.)

Exercise 17 review

// length function, while loop version

```
int length(const Node *n) {
  int count = 0;
  while (n != NULL) {
    count++;
    n = n->next;
  }
  return count;
}
```

Note: "const Node *n" means "n is a pointer to const Node"

We can't change the contents of the node n points to,
but we can change *which* node n points to

Exercise 17 review

```
// length function, recursive version!

int length(const Node *n) {
  if (n == NULL) {
    return 0;
  } else {
    return 1 + length(n->next);
  }
}
```

Observation: a linked list is a recursive data structure.
If a pointer n points to:

- NULL: the list is empty

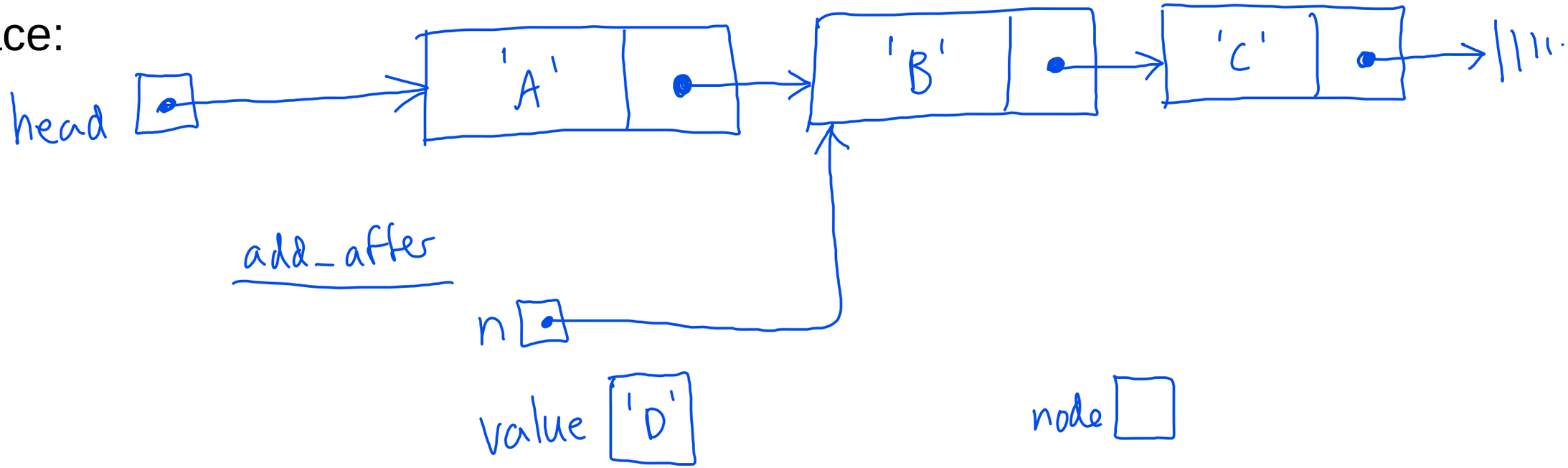
- a Node: the list has at least one element, and n->next points to a smaller linked list

Many important data structures can be defined recursively.

Exercise 17 review

```
void add_after(Node *n, char value) {
  Node *node = (Node *) malloc(sizeof(Node));
  node->data = value;
  node->next = n->next;
  n->next = node;
}
```

Trace:



Exercise 17 review

```
void reverse_print(const Node *n) {
  // Pseudo-code:

  //   if n is the empty list
  //     do nothing, return
  //   else
  //     print the rest of the list in reverse order
  //     print the value of the first element
}
```

Day 18 recap questions

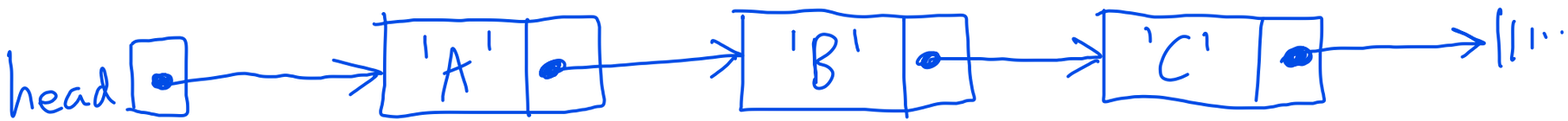
1. How do you implement add_front on a linked list?
2. How do you modify a singly linked list to create a doubly linked list?
3. How do you make a copy of a singly linked list?
4. Why does add_after takes a Node * as input, but add_front takes Node **?
5. What cases should be handled when implementing remove_front?

4. Why does add_after takes a Node * as input, but add_front takes Node **?

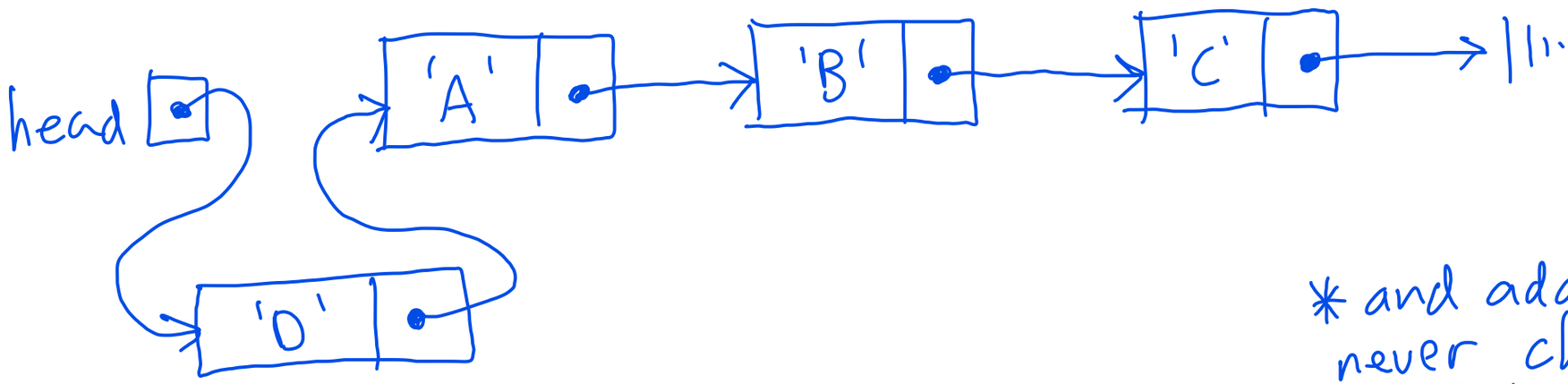
Because add_front needs to change which node the head pointer points to.*

add_front (list head? , 'D')

Before :



After



* and add_after never changes which node the head pointer points to

1. How do you implement add_front on a linked list?

```
void add_front(Node **p_head, char value) {
  Node *node = (Node *) malloc(sizeof(Node));
  node->data = value;
  node->next = (*p_head)->next;
  *p_head = node;
}
```

add_front(&head, 'D')

Trace: