slido.com
jhuintprog

601.220 Intermediate Programming

Summer 2022, Meeting 11 (July 1st)

Today's agenda

- Review of exercises 17 and 18
- Work on midterm project

Reminders/Announcements

- Midterm project (code) is due by 11pm **this evening**
  - Only one team member needs to submit
- Midterm project individual contributions survey: complete by 11pm on **Sunday, July 3rd**
- Midterm exam: in class on **Wednesday, July 6th**
  - Exam details:
    - *Synchonous*, i.e., you *must* attend the Zoom meeting
    - You will work in a breakout room with your camera on
    - Access to internet resources, editor/compiler, etc. is allowed
    - Communication with or help from other people is prohibited
  - Review session: details TBA, but probably 5–6 pm on Tuesday, July 5th

Exercise 17 review

Node data type:

```
typedef struct Node_ {
  char data;
  struct Node_ *next;
} Node;
```

The `typedef` allows us to refer to the "`struct Node_`" type as just "`Node`".

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;
}
```

advance

Note: `const Node *n` means "`n` is a pointer to `const Node`". Function is saying that it won't modify the object that `n` points to.

Exercise 17 review

```
// length function, recursive version
int length(const Node *n) {
  if (n == NULL) {
    return 0;
  }
  return 1 + length(n->next);
}
```

base case

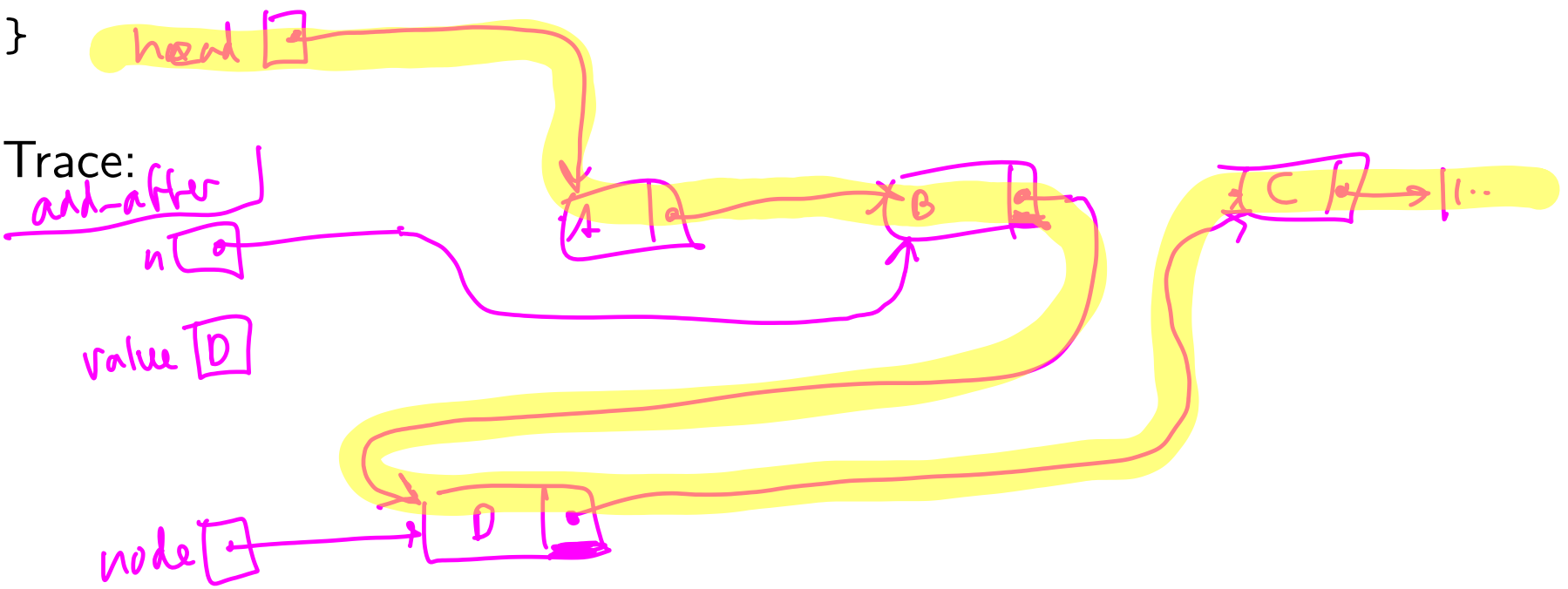
A linked list can be considered as a *recursive* data structure.
Assume `n` is a pointer to a linked list node. Cases:

1. `n` is NULL: the list is empty
2. `n` points to a node: nonempty list, `n->next` points to a smaller list (with one fewer nodes than the overall list)

Exercise 17 review

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

Trace:



Exercise 17 review

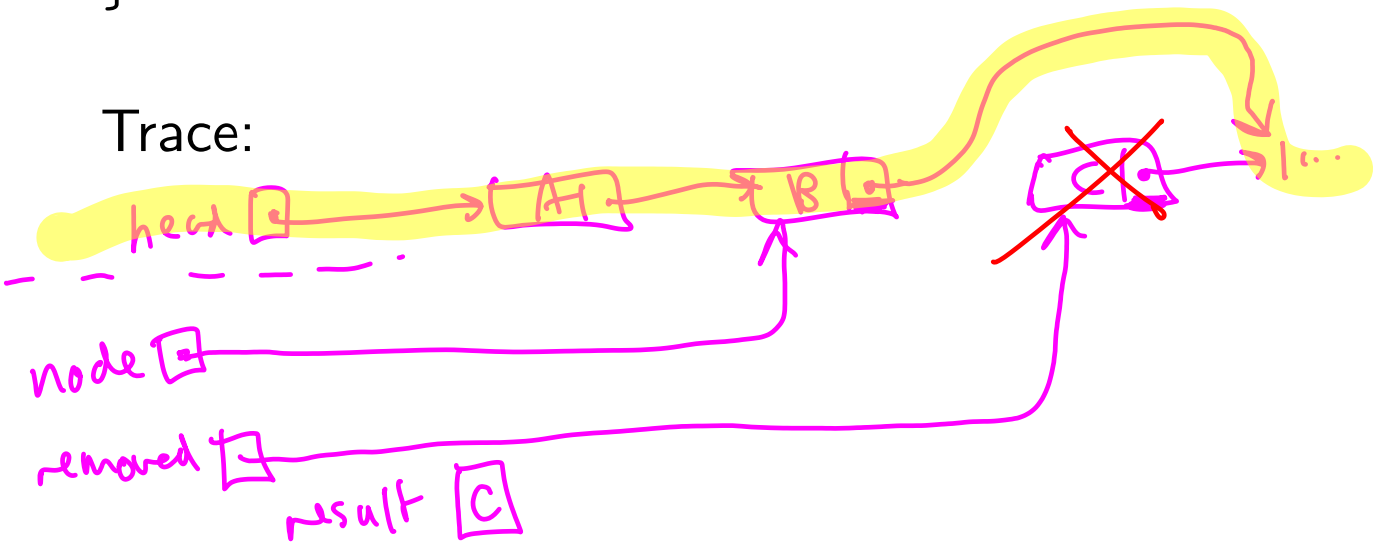
```
void reverse_print(const Struct Node *n) {
  // Pseudo code:
  // if (n is the empty list)
  //   do nothing, return
  // else
  //   print the rest of the list in reverse order
  //   print the value of the first element
}
```

reverse_print(n->next);

Exercise 18 review

```
void remove_after(Node *node) {
  Node *removed = node->next;
  if (removed == NULL) { return '?'; }

  node->next = removed->next;
  char result = removed->data;
  free(removed);
  return result;
}
```

Trace:



Exercise 18 review

```
char remove_front(Node **list_ptr) {
  if (*list_ptr == NULL) { return '?'; }

  Node *removed = *list_ptr;
  *list_ptr = removed->next;
  char result = removed->data;
  free(removed);
  return result;
}
```

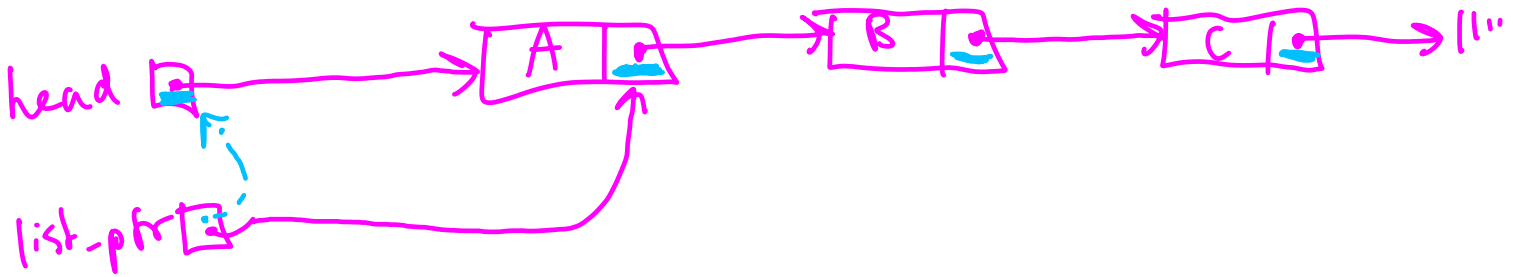
Trace:

Exercise 18 review

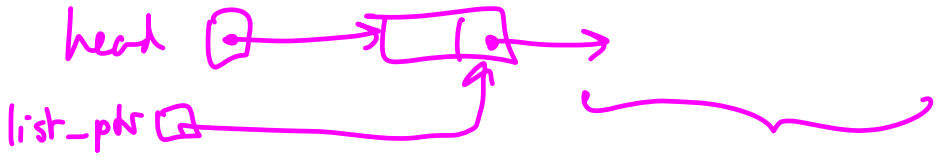
```
void remove_all(Node **list_ptr, char val) {
  if (*list_ptr == NULL) return; // reached end of list?

  if ((*list_ptr)->data == val) {
    // remove first element
    remove_front(list_ptr);
  } else {
    // skip first element
    list_ptr = &((*list_ptr)->next);
  }
  remove_all(list_ptr, val); // remove remaining occurrences
}
```

think about it

head



Exercise 18 review




```
Node *insert(Node **list_ptr, char val) {
  if (*list_ptr == NULL || val < (*list_ptr)->data) {
    add_front(list_ptr, val);
    return *list_ptr;
  } else {
    // recursion
    return insert( &(*list_ptr)->next, val );
  }
}
```

think!

Work on midterm project!

- You can also ask questions about exercises and/or exam review material
- Breakout rooms 1–10 are "social"
- Use Slack to let us know if you have a question
  - This is preferred: the CAs have no way of seeing the Zoom "ask for help" feature

Notes

Notes

Notes

Notes

Notes