601.220 Intermediate Programming

Summer 2023, Meeting 11 (June 30)

Today's agenda

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

Updates about midterm project

- Deadline for midterm project will be moved one day. New deadline is **Saturday July 1st at 11pm**.
- No further extensions will be provided for the project.
- Deadline for the indiviual contributions survey is still **Sunday, July 2nd at 11pm**.
- **Reminder**:Late days cannot be used for the midterm or final project.
- Only one team member needs to submit the code. Everybody fills the individual contributions survey.

Updates about midterm project

Small change in Binarize function instructions

Before: You do not need to check if the input threshold is an integer or not, but you do need to check it is a valid number and is between 0 and 255.

changed to: You need to check if the input threshold is an integer or not, and check it is a valid number between 0 and 255. If not report and error.

* Float values are OK along as they are between 0 and 255.

Notes - More midterm project clarifications

* Make sure to always revise the output of the autograder, even if it shows green

* Rescaled images provided on the starter code are irrelevant for

Reminders/Announcements

- Midterm exam: in class on **Wednesday, July 5th**, see full post in Piazza.
  - 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
  - **NOT** Allowed during the exam:
    - The use of AI-based tools, such as ChatGPT, GitHub copilot, and GitHub copilot chat, is strictly prohibited. Visual studio code will not be allowed during the exam due to its tight integration with GitHub Copilot.
  - Review session: lead by head TA, on Sunday, July 2, 8-9 pm

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

```c
// 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`". 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

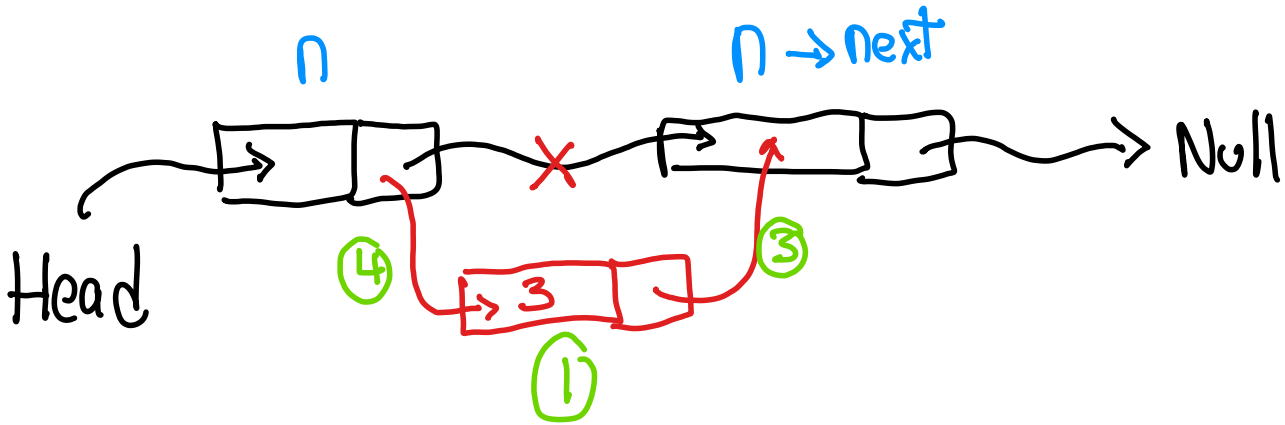
A linked list can be considered as a *recursive* data structure. Assume `n` is a pointer to a linked list node. Cases:

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));   (1)
  node->data = value;  (2)
  node->next = n->next;  (3)
  n->next = node;  (4)
}
```

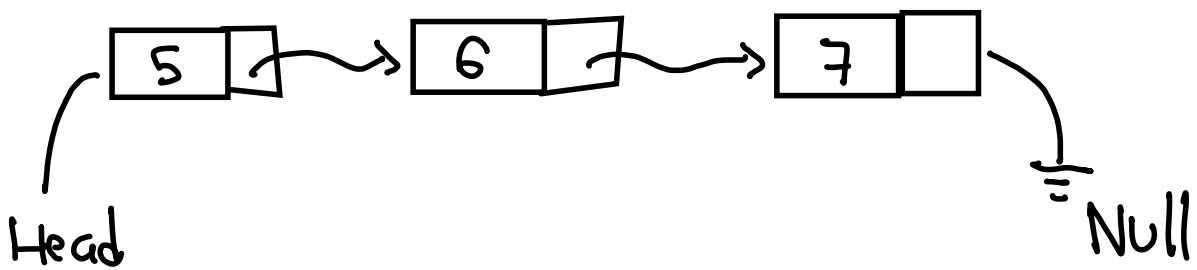
Trace:



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
}
```

- call reverse print again



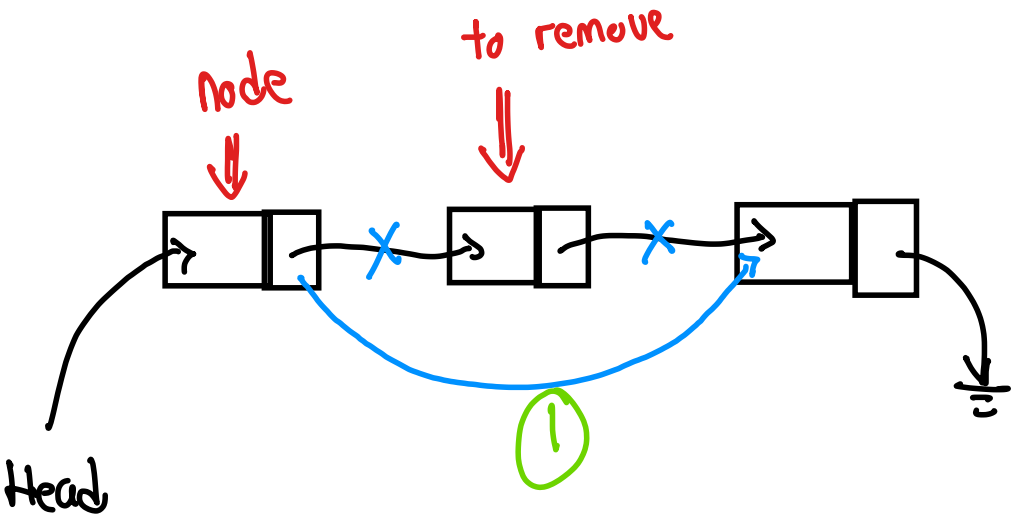
Exercise 18 review

char

```
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:

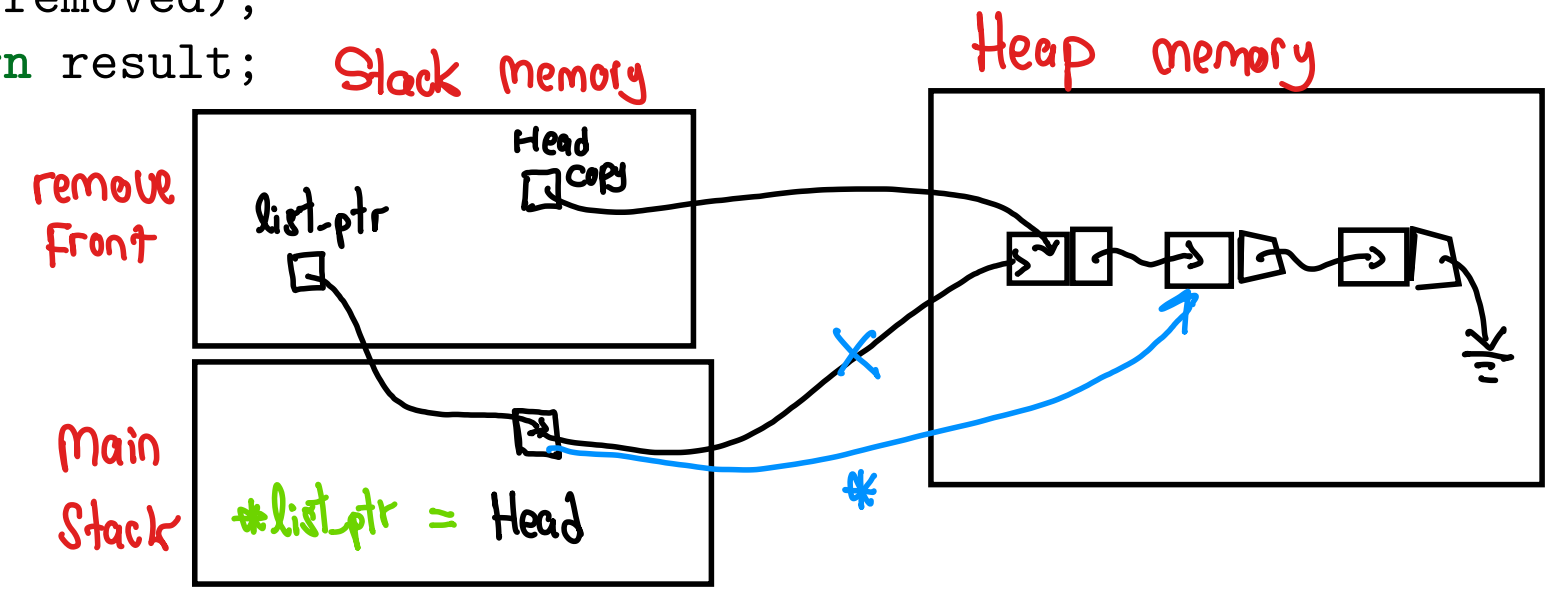


Exercise 18 review

```
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

  } else {
    // skip first element

  }
  remove_all(list_ptr, val); // remove remaining occurrences
}
```

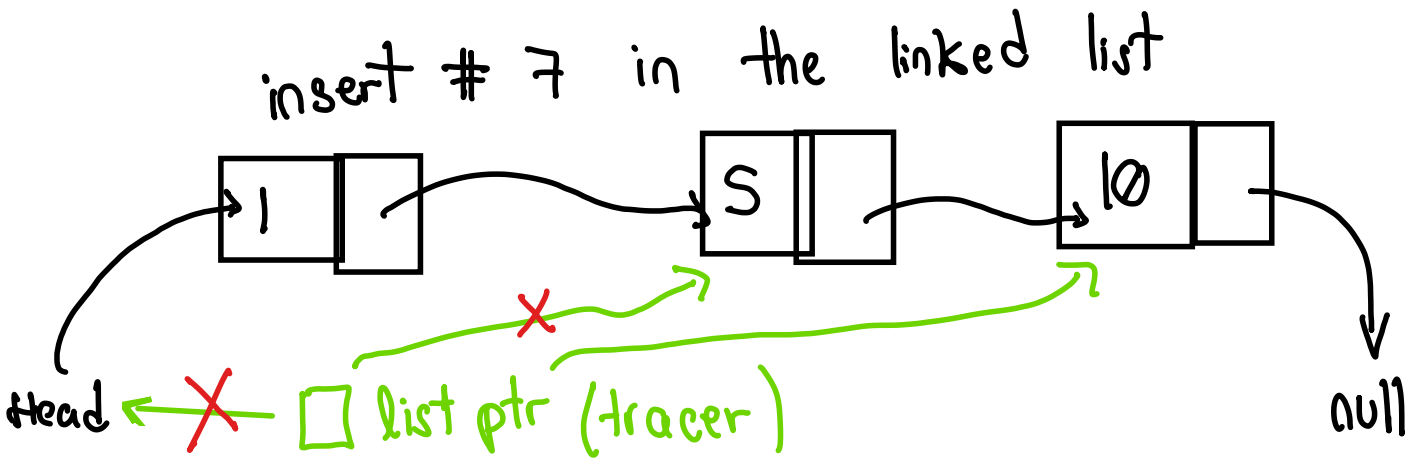
list_ptr = &((*list_ptr)->next);

Exercise 18 review

→ insert while keeping the list sorted.

```
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
    list_ptr = &((*list_ptr)->next);
  }
}
```

⇒ move list_ptr to next node



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