601.220 Intermediate Programming

Spring 2023, Day 22 (March 13th)

Today's agenda

- Exercise 18 review
- Day 22 recap questions
- No exercise (you can work on the midterm project)

Announcements/reminders

- Midterm project due Friday at 11 pm
  - No late submissions
  - Preliminary autograder is available

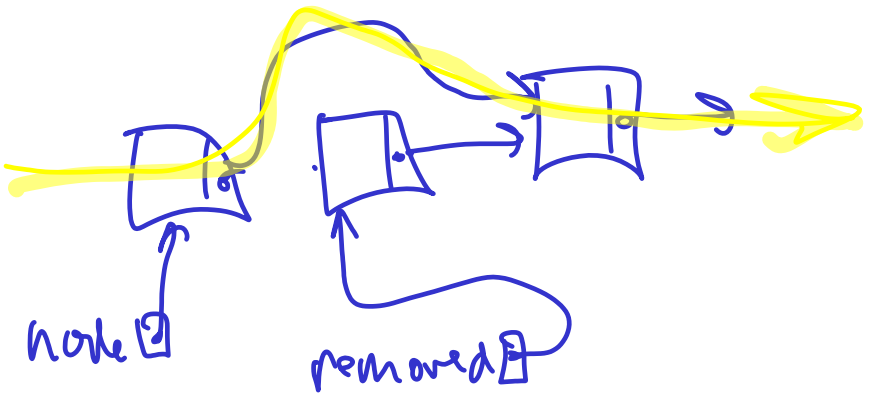
Note: I will not be holding office hours on Thursday 3/16

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:

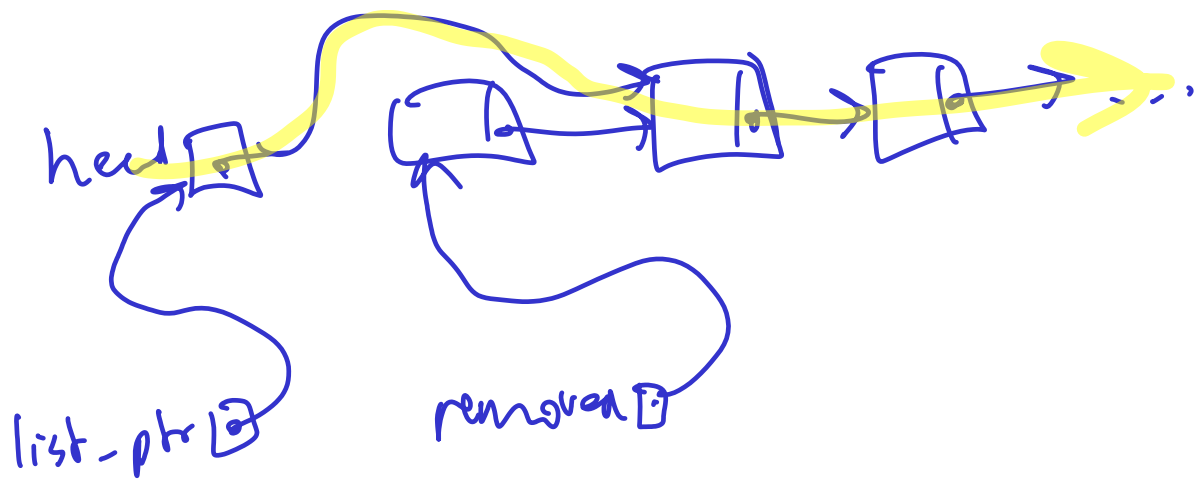


Exercise 18 review

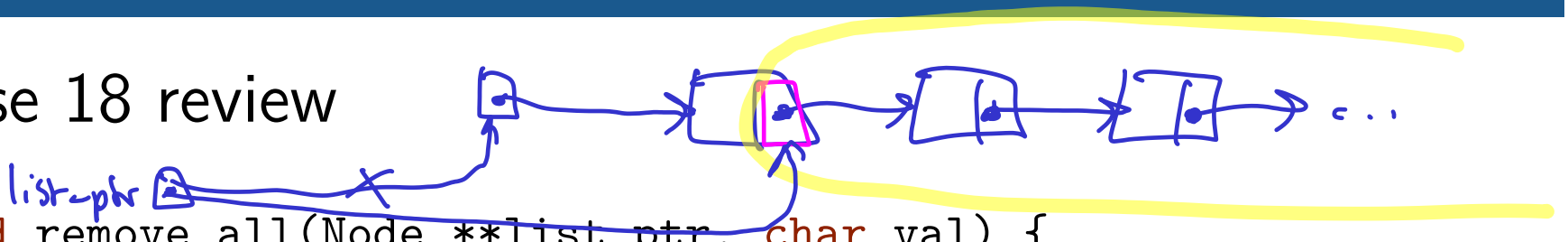
```
char remove_front(Node **list_ptr) {
  if (*list_ptr == NULL) { return '?'; }

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

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);

  }
}
```

1. What is the difference between C and C++?

C: “portable assembly language”

Very little direct support for rich data types.

Very little support for automatic resource management.

No support for generic programming.

Almost no run-time support is needed.

C is a relatively simple language.

C vs. C++

C++: a modern object-oriented language

C++ classes allow the creation of very rich data types.

Constructors and destructors shift the burden of resource management (memory, files, etc.) onto the compiler.

Extensive support for generic programming (template classes and functions.)

Some runtime support (e.g., exceptions) is required.

C++ is arguably the most complex programming language ever created.

2. What is a namespace in C++?

A namespace allows names of classes, functions, data types, etc. to be isolated so that they don't conflict with other classes/functions/data types that happen to have the same name.

E.g., all C++ standard library classes are in the `std` namespace. For example, `std::string` is the `string` class provided by the standard library. You could define your own class called "`string`", and it would not conflict with `std::string`.

If you are developing a library that you intend to be incorporated into other programs, put all of its classes/functions/data types/etc. into a library-specific namespace.

3. Why should you not put "using" statements in header files?

Because you would be forcing any code that `#includes` the header file to accept the name(s) imported by the `using` statement.

For example, if your header does

```
using std::string;
```

then you make it difficult for code `#includeing` your header to use any other class called "`string`".

4. How do you read and write in C++ (i.e. standard inputting/outputting)?

Writing data: use `std::cout`. Reading data: use `std::cin`. You need to `#include <iostream>` to use these.

Example:

```
std::string name;
int age;

std::cout << "What's your name? ";
std::cin >> name;

std::cout << "How old are you? ";
std::cin >> age;

std::cout << "Hello, " << name << ", nice to meet you\n";
```

5. What is the difference between C strings and C++ strings?

C strings:

Stored in an array of `char` elements.

Entirely the programmer's responsibility to ensure the string is not too large for the array.

Dealing with arbitrary-length strings (e.g., reading lines of text from a file) is quite tricky, and will generally involve dynamic allocation and resizing buffers on the fly.

C strings vs. C++ strings

C++ strings:

Storage for the character sequence is handled automatically.

Storage grows as needed! And, is automatically freed when the string is destroyed.

Dealing with arbitrary-length strings is easy.

6. How long can a C++ string be?

As long as it needs to be, limited only be available memory.

Notes

Notes

Notes

Notes

Notes