Recitation#8: C struct and pointer: Linked List Interview

CS232 Spring 2021

When: March 19 at 2:00 pm

Linked List is the best exercise to learn pointers and structures in C

Here's another exercise to help you in your interviews! For this exercise, you will fill in the functions `append_node()` and `reverse_list()` in `list.c`. Don't forget to look at `list.h` to see the definition of the node struct and also to see the general format of a simple .h (header) file!

In `append_node()`, you will append a node to the end of a linked list.

In `reverse_list()`, you will reverse a linked list in place (i.e. without creating a new list). For example, the list `1->2->3->4->5` would become `5->4->3->2->1`. HINT: In each iteration of your loop to traverse the linked list, think about what information you need to store in variables before reversing the link so that you can still successfully keep iterating through the list and reversing links. This is a fairly difficult exercise in C; we recommend that you draw a picture if you get stuck because doing so tends to make double pointers seem less intimidating. Please write up what you think the algorithm should be before asking a TA for help. You can think of the double pointer as simply a pointer to a list of pointers to nodes.

Why do both of these functions take in a `node**` and not a `node*`? Think about memory management; if the input was a node*, would it be possible to modify the pointer that was passed into the function from, say, the main() function? Remember that C is pass-by-value!

ACTION ITEM: Finish implementing `append_node()` and `reverse_list()`.

Once you complete these functions, you can compile it against the test code `test_list.c` and run your code using the following commands. If you make any changes, make sure to run ALL of the following commands again, in order.

```
$ gcc -c list.c
$ gcc -c test_list.c
$ gcc -g -o test_list test_list.o list.o        //This will create an executable by linking test_list.o and list.o.
$ ./test_list
```

Credits: the exercise come from CS61C fall 2019