161-01

Stacks & Queues with arrays

Friday December 6

Agenda

0. Quiz Feedback
1. Announcements
2. Discussion
3. Lab

Quiz Feedback

Announcements

- Homework 8
  - Due Monday evening
  - Questions?

- More candidate talks (token eligible):
  Tuesday Dec 10 at 12pm
  Thursday Dec 12 at 4pm
  Tuesday Dec 17 at 12pm

Wednesday's Lab

1. You were to use a linked list ADT, there was no "`node`" struct in the header, you need to add it

2. we struggled with `pop`! A couple notes:
   - The documentation for pop says that it returns a owning pointer to the string
   - `stack-driver.c` has a line `free(str)` in the pop portion of the code
   - Together, this tells us that we should return a pointer to the string. Pop should free the node but *not* the string
   - If I told you to use `strcpy`, I led you astray

3. You do not need to turn in anything from Wednesday's lab, but you should be comfortable with implementing a stack and queue with a linked list for the exam (and for Homework 8!)

Today’s Lab

1. Lesson learned: Carefully read the documentation for functions to figure out what should be returned and what should be freed

2. Today: Implementing a stack and a queue using a dynamic array
   - You can use your previous code where you implemented a linked-list with a dynamic array to help.
   - Our struct definition (in the .h file) should be a double pointer
   - The `init` function (in the .c file) should use `malloc` to create space for it on the heap
   - Initialize the capacity of your array to be something like 5 or 10.
   - When you need to `realloc`, do not add a single new space
   - You need to decide where in the array is the “start” of your queue/stack, and where is the “end”

3. I’ll have you turn in your stack implementation (using a dynamic array) with this week’s lab report

Lab

Wrap up

For next time:

- Read textbook section 17.7
- Work on Homework 8

The Stack ADT

Think of a stack like a stack of cards, where we can only remove a card from the top of the file.

The last value added is the first one returned (last in, first out)

Interface:
- push: add a value to the top of the stack
- pop: take a value off the top of the stack and return it
- peek: look at the value on the top of the stack without removing it

The Queue ADT

A queue holds values as if they were waiting in line.

The first value added is the first one returned (first in, first out)

Interface:
- add: add a value to the back of the queue (aka “enqueue”)
- take: take a value from the front of the queue (aka “dequeue”)
- peek: look at the value at the front of the queue without removing it