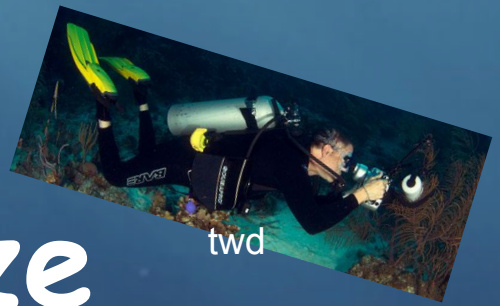
CS33 Project Gear Up



twd

Maze

What is a Gear Up?

A place to address the questions:

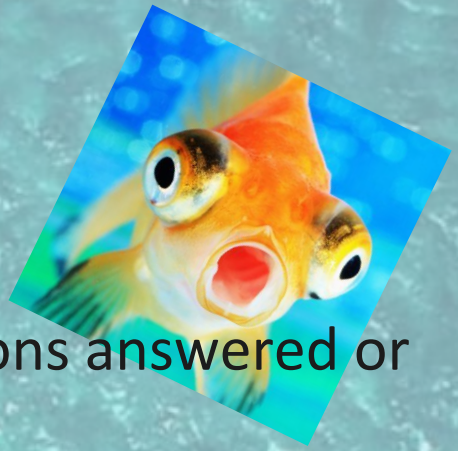
- “how do I start?”
- “help?!??”

A place to ask review and clarify project-related material.

A place to get tips from TAs who have done the assignment.

- We’re here for you!

What is a Gear Up *Not*?



It is not TA hours

- This isn't the time to get code-specific questions answered or get debugging help.
- We won't be talking about grades or addressing any grading concerns.

It is not "Sanctioned Collaboration"

- We encourage you to get to know your classmates, but you don't need help debugging at the beginning of the project.
- You should be asking the TAs your questions, not other students.

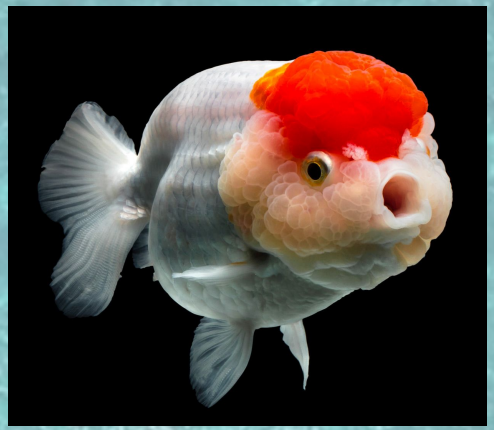
Overview

You will be writing a program that will generate mazes in memory (`generator`), and then another program which solves those mazes (`solver`).

Topics

- C language basics!!
  - Arrays, structs, and enums
  - Pointers
- Bitwise operations

Basic Pointer Review



- int num = 5;

  addr = # ← addr now holds the memory address of num

  val = *num; ← val holds the value of num, 5
- int arr[10]
  - arr refers to the memory address of the first element of the array

Bitwise Review

You will use these bitwise operations when encoding the maze into memory.

- AND (1 if both bits are 1): 1011 & 0110 = 0010
- OR (1 if either bit is 1): 1011 | 0110 = 1111
- NOT (flip 1's and 0's): !(1011) = 0100

Roadmap

`cs0330_install maze`

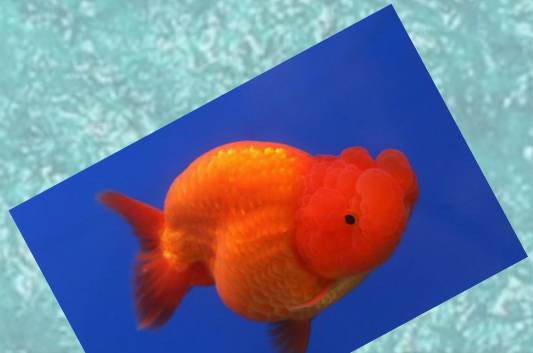
1. Start filling in helper functions in common.c
    a. This file contains functions that will be used in implementing both your generator *and* solver.
2. Understand the *drunken walk* pseudocode in the handout and implement it to generate the maze.
3. Test your generator thoroughly before moving on to solver.
4. Follow a similar procedure for solver using the given *dfs* pseudocode!

Demos!

Generator, Solver, Validator

Tips

- Your program shouldn't all be in `main()`.
- If you get a segmentation fault (segfault) run "backtrace" (bt) in gdb to see where the problem happened
- Functions that make system calls can return error values. Make sure to handle them! (use fprintf)
- Use header files to map out shared structures, and use C files to implement them.
- Learn to use the man pages! They're awesome and very helpful.
  - ex. `$ man 3 printf`
  - ex. `$ man 2 kill`
  - when in doubt, `$ man man` ;)



Tips, Cont'd

- Make sure to read and follow the style guide on the class website.
- If your code isn't working, try looking at your pointers.
  - Pointers are a tricky concept, and can cause problems if not implemented exactly.
- If you get a segfault, look at where you access arrays.
  - Segfaults commonly occur when you try to access "out of bounds" memory.
  - Your program should not segfault for any reason, even invalid input.
- **Use the CS33 Course Calendar!**

So Why Am I Doing This?

- To learn basic C syntax
- To learn how to structure C programs
- Learning to program in C will help you understand how computer memory works!
- Your grade depends on this assignment

Questions?

Ask anything about the assignment,
the topics the assignment covers,
or anything else!