



LAB 03:

Arrays and Swap

Provided Files

- `main.c`
- `myLib.c`
- `myLib.h`
- `cat.c`
- `cat.h`

Files to Edit/Add

- `main.c`
- `cat.c`
- `Makefile`
- `.vscode`
 - `tasks.json`

Instructions

In this lab, you'll be writing code to swap cat structs. Note: Make sure to copy over your `Makefile` and `.vscode/tasks.json` from one of your previous assignments.

TODO 1.0

- In `cat.c`, initialize the `catBitmaps` array. The array should contain the already defined bitmaps: `bitmap1`, `bitmap2`, `bitmap3`, `bitmap4`, and `bitmap5`.

TODO 2.0

- In `cat.c`, complete the switch statement. In this switch statement, we are setting pixels a particular color based on the retrieved value from the cat bitmap. Each cat bitmap has 6 possible values, 0, 1, 2, 3, 4, or 5. Create a [case](#) for values 1, 2, 3, 4, and 5. The 0 case can be handled by the default statement, as we will not set the pixel any color if the value of the bitmap is 0.
 - For case 1, set the color of the pixel at `col + i`, `row + j` to `WHITE`.



- `WHITE` a macro defined for you in `myLib.h`.
- For case 2, set the color of the pixel to `MAGENTA`.
 - `MAGENTA` is a macro defined for you in `myLib.h`.
- For case 3, set the color of the pixel to `GREY`.
 - `GREY` is a macro defined for you in `myLib.h`.
- For case 4, set the color of the pixel to the cat's `furColor`.
 - Make use of the cat pointer that is passed into the function, and access that cat struct's `furColor` member.
- For case 5, set the color of the pixel to `LIGHTGREY`.
 - `LIGHTGREY` is a macro defined for you in `myLib.h`.
- Build and run your lab thus far. You should see the following. If you do not, fix your code before moving forward.



TODO 3.0-3.2

This requires several moving parts, so it is broken into three parts.

- TODO 3.0
 - At the bottom of `main.c`, write a function called `swap` that swaps two cat structs. Refer to lecture material to get a refresher on how to implement swap if needed.
- TODO 3.1
 - At the top of `main.c`, add the function prototype for the swap method you just wrote.
- TODO 3.2
 - In `main.c`, implement the `reverseCats` function. To do this, you will write an in-place array reversal for the `cats` array. "In-place" means that you may not use an additional array or data structure to reverse the contents of the array. Thus, you will need to call the `swap` method you wrote to perform the in-place array reversal.
 - **Hint:** you only need to iterate through half of the array to swap the elements!
 - **Hint:** there's no `array.length` method in C! Make use of the



CATSCOUNT macro in `cat.h` instead.

- Build and run your lab. When you press START, you should see that the cats have reversed their order. You can press START again to see that the cats have returned to their original positions.
 - The images that follow show what you should see (1) before you've reversed your cats and (2) after you've reversed your cats. If you see (1), then (2) after pressing START, you've completed the lab correctly. Otherwise, fix your code before moving forward.

(1)



(2)



You will know if it runs correctly if you can:

- Press START to see the cats reversed

Tips

- Review lecture materials for how to implement swap.
 - Draw out what your logic for `reverseCats` actually does. This will help you ensure you've implemented the code correctly!
 - Follow each TODO in order, and only move forward if everything is correct
-

Submission Instructions

Zip up your entire project folder, including all source files, the Makefile, and everything



produced during compilation **(including the .gba file)**. Submit this zip on Canvas. Name your submission Lab03_FirstnameLastname, for example: "Lab03_KittyKat.zip".