Lab 5
General Comments
Pushing tested code to your Git repo

- **IMPORTANT:**
  - You must **compile** and **test** your code before pushing it to your Git repo
  - **Why is this important?**
Lab 5
Incremental Development
Incremental Development

- **Idea:** develop your program incrementally, a “chunk” at a time
- **Why?** So that, if the “chunk” is faulty, you know where to look for the bug(s) -> the “chunk”
- **“chunk” can be:**
  - Function(s)
  - Class
  - Feature
  - Etc...
- **Process:**
  - Once you have designed (algorithm) and implemented (code) the “chunk”, you compile it then test it using a test driver -> main()
  - Only once the chunk works (not only compiles but actually “solves the problem”) one can move on to the next “chunk”
Incremental Development in Lab 3!

- Lab 3 (imgops.c) well set up for incremental development
- Why?
  - imgops.c already has stubs
  - These stubs allow imgops.c to compile without adding our code yet
  - So, we can design, implement, compile and test each function one at a time
  - Grading robot grades one function at a time as well

```c
/*
 * PART 1: OPERATIONS ON THE WHOLE IMAGE
 */

/* TASK 1 - Easy functions to get started */

void zero( uint8_t array[],
           unsigned int cols,
           unsigned int rows )
{
    // your code here.
}

// Set every pixel to 0 (black)

// This is a function stub.
// This stub does not need to return anything because this function is a void function.

uint8_t* copy( const uint8_t array[],
               unsigned int cols,
               unsigned int rows )
{
    // your code here
    return NULL;
}

// Returns a pointer to a freshly allocated array
// same values as the original array, or a null
// allocation fails. The caller is responsible
// later.
```

More info: https://en.wikipedia.org/wiki/Method_stub
Incrementally Developing Lab 5

- Lab 5 (intarr.h) needs to be set up for incremental development
- How?
  - Create intarr.c
  - Initially implementing each function as stubs
    - This allows intarr.c to compile without our code
  - Then, we can design, implement, compile and test each function one at a time
Lab 5
Helpful Tips
Using the “Multi Source Files” model

- **intarr.c**: This file contains definition (body) of some functions.
- **testDriver.c**: This file contains the main function and possibly others functions (optional). It may call functions defined in other files.
- **intarr.h**: This file contains function headers (also called function declarations, or function prototypes).
Introducing `struct`
Helpful Tips about Lab 5

- `free(aPtr);` should be followed by `aPtr = NULL;`
  - Unless you are using `free` at the end of your program

- Useful functions:
  - `malloc()` + `free()`
  - `memcpy()`
  - `realloc()` (may be useful in `resize()`)

- Do not forget to increase `len` after a successful call to `realloc()`
Helpful Tips about Lab 5

You may want to investigate ...

- the function `assert()`
  - How it works
  - What it returns
- `enum`