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 */

// Set every pixel to 0 (black)
void zero( uint8 t array[],
           unsigned int cols,
           unsigned int rows )
{
    // your code here.
}

// Returns a pointer to a freshly allocated array
// same values as the original array, or a null
// allocation fails. The caller is responsible
// later.
uint8_t* copy( const uint8_t array[],
               unsigned int cols,
               unsigned int rows )
{
    // your code here
    return NULL;
}
```

This is also a function stub. However, this stub does need to return something to satisfy the function declaration.

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

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**
  - C code
  - This file contains definition (body) of some functions.

- **testDriver.c**
  - C code
  - Includes `intarr.h`
  - This file contains the main function and possibly others functions (optional). It may call functions defined in other files.

- **intarr.h**
  - Header
  - This file contains function headers (also called function declarations, or function prototypes).
Introducing *struct*

- **struct**
  - *ia*
  - *intarr_t*
    - *data*
    - *len*
  - Memory for this struct needs to be allocated

- **array**
  - array of *len* int's
  - Memory for this array needs to be allocated

- **array of width x height of unint8_t**
  - *img*
  - *width*
  - *height*

*len* is the size of the array data and the number of elements in data
Helpful Tips about Lab 5

- `free( aPtr );` should be followed by `aPtr = NULL;`
- Useful functions:
  - `malloc( ) + free( )`
  - `memcpy( )`
  - `realloc( )` (may be useful in `resize( )`)
- Do not forget to modify `len` after a successful call to `realloc( )`
- Always validate the parameters to functions
Helpful Tips about Lab 5

You may want to investigate ...

- the function `assert( )`
  - How it works
  - What it returns
- `enum`
Task 6 and Task 7

- Do Task 7 before Task 6
- In Task 6
  - In \texttt{intarr\_push(...)} -> it makes total sense to call \texttt{intarr\_resize(...)}
  - In \texttt{intarr\_pop(...)} -> Careful calling \texttt{intarr\_resize(...)}
    - \textbf{Why?} What happens when you are pop’ing the last element?
      - \texttt{realloc} using size 0 -> problematic (unpredictable)
    - Check it out: \url{https://en.cppreference.com/w/c/memory/realloc}

- Solutions:
  1. You may choose not to call \texttt{intarr\_resize(...)} in \texttt{intarr\_pop(...)}
  2. You may add an “if” statement in \texttt{intarr\_resize(...)} to avoid calling \texttt{realloc} when len = 0