Lab 6
Helpful Tips
In general – for all labs/functions: Are Function Parameters Valid?

- Always check the validity of a function’s parameters
  - Are they NULL?
  - Are their values “out of range”?
- Once validated, parameters can be used safely!
- Example 1 from Lab 5:

```c
// Frees all memory allocated for ia. If the pointer is null, do nothing. If the ia->data is null, do not attempt to free it.
void intarr_destroy( intarr_t* ia )
{
    // If the pointer is null, do nothing.
    if ( ia == NULL )
        return;

    // If the ia->data is null, do not attempt to free it.
    if ( ia->data ) { // ia->data != NULL
        // Frees all memory allocated for ia->data.
```
In general – for all functions: Are Function Parameters Valid?

Example 2 from Lab 6:

```c
/* LAB 6 TASK 1 */

/*
Save the entire array ia into a file called 'filename' in a binary file format that can be loaded by intarr_load_binary(). Returns zero on success, or a non-zero error code on failure. Arrays of length 0 should produce an output file containing an empty array.

Make sure you validate the parameters before you use them.
*/

int intarr_save_binary(int* ia, const char* filename) {
    if (ia == NULL) {
        //puts( "intarray is NULL" );
        return 1;
    }

    if (filename == NULL) {
        //puts( "Filename is NULL" );
        return 2;
    }

    /* Implementation of file saving goes here */
}
```
In general – for all functions: Is Function’s Returned Value Valid?

- Always check the validity of the value returned by a function
  - If the function returns a pointer: is it NULL?
  - If the function returns a value: is it a value we are expecting?
- Once validated, the returned value can be used safely!
In general – for all functions: Is Function’s Returned Value Valid?

Example:

```c
/* LAB 6 TASK 1 */

/*
   Save the entire array ia into a file called 'filename' in a binary file format that can be loaded by intarr load binary(). Returns zero on success, or a non-zero error code on failure. Arrays of length 0 should produce an output file containing an empty array.

   Make sure you validate the parameters before you use them.
*/

int intarr_save_binary( intarr_t* ia, const char* filename )
{
...

   FILE* f = fopen( filename, "w" );
   if( f == NULL )
   {
      //puts( "Failed to open array file for writing" );
      return 3;
   }

   if( fwrite( &ia->len, sizeof(unsigned int), 1, f ) != 1 )
   {
      //puts( "Failed to write array length header" );
      return 4;
   }

   return 0;
}
```
Content of files

Task 1

- save

Task 2

- save

binary file

- len
- array
- load

json file

- \n
- #, \n
- #, \n
- ...

- ...

- ...

- ...

- ...

- \n
- string of characters
- load

Each arrow represents one “write” to file

Each arrow represents one “read” from file

load

optional
Explaining some of the Requirements

- Task 1 – Requirement 5
  - 1 call to `fwrite()` for the whole array of #’s

- Task 2 – Requirement 3
  - Within a `for` loop, you can call `snprintf()` and `fwrite()` for each number (#) separately
For both Tasks of Lab 6

- Make use of some of the functions you implemented in Lab 5 “intarr.c”

- How would you test your tasks?
  - Create a “TestDriver.c”