Homework 2: Interfaces, Implementations, and Images
By: Nicholas Flores

UArray2:

1. The things to be represented through abstraction in this assignment are the indices of a two-dimensional array, or in the more specific sudoku case, the value and location of the spots in the array.
2.
    - *T UArray2_new (int width, int height, int size)* - instantiates the 2D array by allocating for its size with width * height * size
    - *void UArray2_free (T *uarray2)* - deallocates the memory used by the array
    - *int UArray2_width (T *uarray2)* - takes in 2D array and returns width
    - *int UArray2_height (T *uarray2)* - takes in 2D array and returns heightß
    - *void *UArray2_at (T uarray2, int row, int column)* - checks a spot in the 2D array given passed in indices
    - *void UArray2_map_col_major (T *uarray2, void apply(int i, int j, T uarray2,void *p_row, void *p_col), int check)* - traverses through the 2D array in a column abiding to column prioritization, apply is used to act as a callback function
    - *void UArray2_map_row_major (T *uarray2, void apply(int i, int j, T uarray2,void *p_row, void *p_col), int check)* - traverses through the 2D array in a row abiding to row prioritization, apply is used to act as a callback function
3.
    - example_array = UArray2_new (width, height, size);
    - UArray2_free (&example_array);
    - int example_width = UArray2_width(example_array);
    - int example_height = UArray2_height(example_array);
    - mark = UArray2_at (example_array, row, column);
    - UArray2_col_major (example_array, callback_func, check);
    - UArray2_row_major (example_array, callback_func, check);
4. This program will be represented by a single type T struct to create the unboxed object allocation, meaning pointees are part of the container. This also implies that when the container dies, so does its contents. The invariants to hold true for this program are:
    - index is within bounds of the 2D array
    - if graymap file represents solved sudoku
    - conditions of solved sudoku puzzle
5. In the sense of "world ideas," this representation will correspond to the aligning of locations in a grid-like fashion, and then touching each location by row/grid prioritization.
6.
    - First test with provided helper code to replicate output
    - Adjust and test with graymap files
    - Write and test with sudoku predicate program
    - Valgrind testing for ^

7.
    - Idiom for allocating and free blocks of memory
    - Idiom for initializing for structs of type T
    - Idiom for usage of apply()

<u>Bit2</u>:

1. The things to be represented through abstraction in this assignment are the indices of a two-dimensional array, or in the more specific black-edged image case, the value (0 or 1) and location of the spots in the array.
2.
    - *T Bit2_new (int width, int height)* - instantiates a 2D array by allocating for its size with width * height.
    - *void Bit2_free (T *bit2)* - deallocates the memory used by the array
    - *int Bit2_width (T *bit2)* - takes in 2D array and returns width
    - *int Bit2_height (T *bit2)* - takes in 2D array and returns height
    - *int Bit2_put (T *bit2, int row, int column, int mark)* - takes in array, index of row and column, and mark to assign to the location passed in
    - *int Bit2_get (T *bit2, int row, int column) -* returns the 0 or 1 bit value at passed index
    - *void Bit2_map_row_major (T *bit2, void apply(int i, int j, T bit2, void *p), int check)* - traverses through the 2D array in a row abiding to row prioritization, apply is used to act as a callback function
    - *void Bit2_map_col_major (T *bit2, void apply(int i, int j, T bit2, void *p), int check)* - traverses through the 2D array in a column abiding to column prioritization, apply is used to act as a callback function
3.
    - example_array = bit2_new (width, height);
    - Bit2_free (&example_array);
    - int example_width = Bit_width(example_array);
    - int example_height = Bit_height(example_array);
    - Bit2_put (example_array, row, column, 0);
    - int mark = Bit2_get (example_array, row, column);
    - Bit2_col_major (example_array, callback_func, check);
    - Bit2_row_major (example_array, callback_func, check);
4. This program will be represented by a single type T struct to create the unboxed object allocation, meaning pointees are part of the container. This also implies that when the container dies, so does its contents. The invariants to hold true for this program are:
    - index is within bounds of 2D array
    - if a pixel is black edge pixel (see black edge pixel conditions)
5. In the sense of “world ideas,” this representation will correspond to the aligning of locations in a grid-like fashion, and then touching each location by row/grid prioritization.
6.

    - First test with helper code to replicate output
    - Adjust and test with image files
    - Test to ensure program knows which pixel is black or not
    - Test with black edged images
    - Test with no black edged images

7.
    - Idiom for allocating and free blocks of memory
    - Idiom for initializing for structs of type T
    - Idiom for usage of apply()
    - Idiom for usage of put() and get()