# <u>iii Design Doc</u>

## Part A

To represent a UArray2 T as a single UArray T
Consists of Interface, Implementation, Key Functions

Interface:

```
#ifndef UARRAY_INCLUDED
#define UARRAY_INCLUDED

#define T UArray
typedef struct T *T;

#undef T

#endif
```

Implemention:

- Create struct T with properties of int width, height, size and UArray rows

extern T     UArray_new(int height, int width, int size);

- Parameters: integers representing the intended height, width, and height of the array the client intends to create
- Purpose:
  Using the provided Hanson UArray interface we hope to build out a 2d from an unboxed 1d array. We intend to call the Hanson new function twice and pass in the width and height as the length making 1 array for the width and x = height array for the height. We then will have the width array point to each of the height arrays we make.
  To do this we will have a while loop to create pointers to allocate enough space for the width.
- Exceptions:
  Will raise an exception if any of the parameters are null or if there is an issue allocating
space

extern int   UArray_height(T  uarray);

- Parameters: T urray
- Purpose:
  Takes in the array and uses the Hanson length function to return the length of the array.
- Exceptions:
- Will raise an exception if any of the parameters are null or if the array is empty (in this case will return 0 but not necessarily quit the program)

extern int   UArray_width(T  uarray);

- Parameters: T uarray
- Purpose:
  Takes in the array and goes into the first element to then use the Hanson length function to return length of the width array (at the 0 index as the width there will be the same everywhere else) which each element in the height array points to.
- Exceptions:
  Will raise an exception if any of the parameters are null.

extern *int   UArray_size  (T  uarray);

- Parameters: T uarray
- Purpose:
  Takes in the array and returns the height and width. Will call the width and height array to get those values and return them in a integer array.
- Exceptions:
  Will raise an exception if any of the parameters are null

extern void *UArray_at    (T  uarray, int row, int col);

- Parameters: the array and the integers corresponding to the intended row or column to be found
- Purpose:
  Takes in the array and finds the element at a specific index. We hope to use the Hanson at function to find the column first and then go to that pointer to find the row correspondence to then find said element.

extern void UArray_map_row_major(T uarray, UArray_apply(), int status)

- Parameters: the array, a pointer function, and a status indicator (delinated by an integer 0 for false and 1 for true)
- Purpose:
  This function will take in the array and then use the map function to map the apply function to each element in the array in row major order. The status will also help us determine if the apply function was applied correctly.
  –Aside: We are thinking that our new function will have to be slightly changed to represent the row major configuration (i.e. have the rows of the array be the main array and then the height be what is pointed to).
- Exceptions:
  Will raise an exception if any of the parameters are null.

extern void UArray_map_col_major(T uarray, UArray_apply(), int status)

- Parameters: the array, a pointer function, and a status indicator
- Purpose:
  Similar purpose as the row major, but we will not have to tweak our new function as the height has already been determined as the main array.

extern void UArrayFree(T *urray);

- Parameters: the array

- Purpose:
  Frees up the memory allocated for the array. We hope to use the Hanson free function to go through the width array and then the height array so no pointers are lost in memory.
- Exceptions:
  Will raise an exception if there is still memory allocation somewhere or if the array is null.

Functions:

- To Search Use: void *UArray_at    (T  uarray, int i);
- To Free: void UArrayFree(T *urray);

## Part B

Similar to part A, represent a 2D array of bits

- Consists of Interface, Implementation
- To Search Use: int *Bit_get  (T  bitArray, int row, int col);
- To Free: void Bit_free(T *bitArray);

Interface:

```
#ifndef BIT2_INCLUDED
#define BIT2_INCLUDED

#define T Bit2_T
typedef struct T *T;

#undef T
#endif
```

Implementation:

- Create struct with properties of width, height and bitArray T rows

extern T    Bit2_new (int width, int height);

- Parameters: integers representing the height and width of the array the client intends to create
- Purpose: Allocate and returns a new bit array
- Exceptions: Will raise an exception if any of the parameters are null or if there is an issue allocating

extern int  Bit2_height(T bitArray);

- Parameters: T bitArray
- Purpose: Takes in the array and uses the Hanson function to return the height of the array as an int.

- Exceptions: Will raise an exception if any of the parameters are null or if the array is empty (in this case will return 0 but not necessarily quit the program)

extern int  Bit2_width(T bitArray);

- Parameters: T bitArray
- Purpose: Takes in the array and uses the Hanson function to return the width of the array as an int.
- Exceptions: Will raise an exception if any of the parameters are null or if the array is empty (in this case will return 0 but not necessarily quit the program)

extern int  Bit2_put(T bitArray, int row, int col, int bit);

- Parameters: T bitArray, int row, int column, int bit
- Purpose: Takes in the array puts the bit at the given row and col and returns the previous value of that bit
- Exceptions: Will raise an exception if any of the parameters are null or if the array is empty (in this case will return 0 but not necessarily quit the program)
- It is a checked runtime error for n to be negative or to be equal to or greater than the length of set, or for bit to be other than zero or one.

extern int  Bit2_get(T bitArray, int row, int col);

- Parameters: the array and the integers for row or column to search
- Purpose: Search the bitArray and finds the element at a specific index as well as tests if there is a value at that index.

extern void Bit2_free(T *bitArray);

- Parameters: the bitArray
- Purpose: frees allocated memory in the bitArray

extern void Bit2_map_row_major(T bitArray, void apply(int width, int height, int bit, void *cl), void *cl);

- Parameters: the bitArray, a void apply function, and a void pointer of closure
- Purpose: Traverse the 2D bit array in row major.  Function maps apply() by iterating over the row by looping over the height of the bitArray
- Exceptions: Will raise an exception if any of the parameters are null.

extern void Bit2_map_col_major(T bitAarray, void apply(int width, int height, int bit, void *cl), void *cl);

- Parameters: the bitArray, a void apply function, and a void pointer of closure
- Purpose: Traverse the 2D bit array in col major order and maps apply().  Function maps and iterates over the column by looping over the height and width of the bitArray
- Exceptions: Will raise an exception if any of the parameters are null.

Part C

TO DO:

- Sudoku checker

Interface:
Implementation:
Functions:
Memory:

Part D

TO DO:

- Removal of Black Edges

Interface:
Implementation:
Functions:
Memory: