UArray2.h

```
#ifndef UARRAY2_INCLUDED
#define UARRAY2_INCLUDED
#define T UArray2_T
typedef struct T *T;

T UArray2_new(T t, int width, int height, int size);

void UArray2_free(T t);
int UArray2_length(T t);
int UArray2_width(T t);
int UArray2_size(T t);
void * UArray2_at(T t, int x, int y);
void UArray2_map_row_major(T t, void apply(int col, int row, T a, void *cl), void
*cl);
void UArray2_map_col_major(T t, void apply(int col, int row, T a, void *cl), void
*cl);
#undef T
#endif
```

BIT2.h

```
#ifndef BIT2_INCLUDED
#define BIT2_INCLUDED
#define T Bit2_T
typedef struct T *T;


void Bit2_new(T t, int width, int height, int size);

void Bit2_free(T t);
int Bit2_length(T t);
int Bit2_width(T t);
int Bit2_size(T t);
int Bit2_get(T t, int n);
int Bit2_put(T t, int n);
void Bit2_map_row_major(T t, void apply(int col, int row, T t, int b, void *cl), void
*cl);
void Bit2_map_col_major(T t, void apply(int col, int row, T t, int c, void *cl), void
*cl);

#undef T
#endif
```

Function Contracts:

UArray2_New():

Description:

Initializes a new 2d unboxed array with size width*height*size and returns the array.

Input:

width – width of the 2d array to create
height – height of 2d array to create
size – number of bytes in each element of the array

Expectations:

Width and height are not negative
size > 0

Returns:

new initialized 2d array.

Exceptions:

Could raise mem_failed error if not enough memory could be allocated.

UArray2_Free():

Description:

Frees the memory of the elements in the 2d unboxed array and sets them to null as well as the memory of the UArray2 itself.

Input:

T – 2d array to free.

Expectations:

Expects that the array passed in is not null
The address of the array passed in is the same as the instance of the UArray2.

UArray2_height():

Description:

Gets the height of the UArray2 which is the number of rows in the UArray2.

Input:

t – UArray2 type

Returns:

Integer value for the height of parameter array

Expectations:

Expects that the array passed in is not null
The address of the array passed is the same as the instance of the UArray2.

UArray2_width():

Description:
Gets the width of the UArray2 which is the number of columns in the UArray2.

Input:
t – UArray2 type

Returns:
Integer value for the width of parameter array

Expectations:
Expects that the array passed in is not null
The address of the array passed is the same as the instance of the UArray2.

UArray2_Size():

Description:
Returns the size of an element in the UArray2, the number of bytes it takes.

Input:
t – UArray2 type

Returns:
Integer value for the size of each element of the parameter UArray2

Expectations:
Expects that the array passed in is not null
The address of the array passed is the same as the instance of the UArray2.

UArray2_At():

Description:
Returns a pointer to the element at the 2d index (col, row).

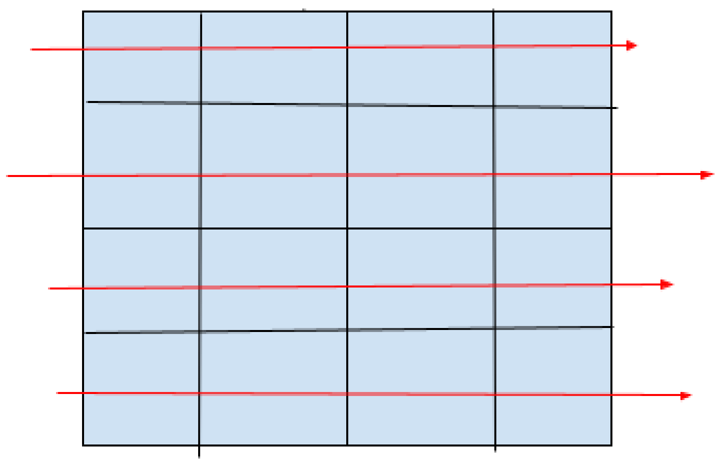
Input:
t – UArray2 type
row – integer for the row to index element
column – integer for the column to index element

Returns:
Void pointer to the element at the desired (col, row) location on the parameterized UArray2

Expectations:
Expects that the indexes given are not out of range [0, width), [0, height)
Expects that the UArray2 passed in is not null
Expects that the address of the UArray2 passed is the same as the instance of the UArray2.

UArray2_map_row_major():

Description: Applies a function to each element of the UArray2 by going across columns first, hitting every element in the row and going down by row.



Input:

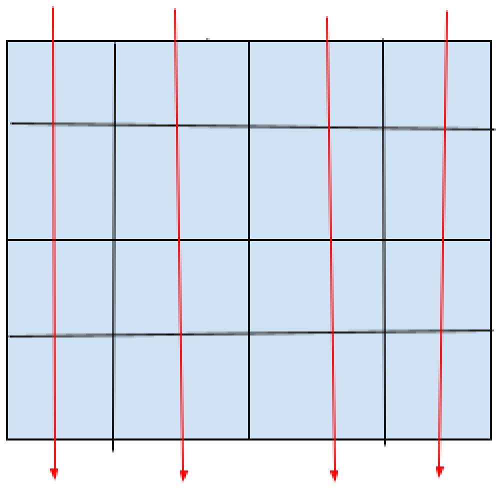
- t – UArray2 type
- An apply function, to perform on each element, whose parameters are:
  - a 2d index: (col, row), t – UArray2 type, a void pointer to the current element, and a closure variable to pass information between each call.
- the closure variable - used to pass extra information between each iteration.

Expectations:

- Expects that the UArray2 passed in is not null
- Expects that the apply function is not null
- Expects that the address of the UArray2 passed is the same as the instance of the UArray2.

UArray2_map_col_major():

Description: Applies a function to each element of the UArray2 by going across rows first, hitting every element in the column and across by column.



Input:

- t – UArray2 type
- An apply function, to perform on each element, whose parameters are:

a 2d index: (col, row), t – UArray2 type, a void pointer to the current element, and a closure variable to pass information between each call.

the closure variable - used to pass extra information between each iteration.

Expectations:

Expects that the UArray2 passed in is not null

Expects that the apply function is not null

Expects that the address of the UArray2 passed is the same as the instance of the UArray2.