```c
#ifndef UARRAY2_INCLUDED
#define UARRAY2_INCLUDED

#define T UArray2_T
typedef struct T *T;

/********** UArray2_new ********
 *
 * Allocates, initializes, and returns a new 2d array of width col and
length
 * row where each element occupies size bytes
 *
 * Parameters:
 *      int col: the number of columns of the 2d array
 *      int row: the number of rows in the 2d array
 *      int size: the number of bytes each element will occupy
 *
 *
 * Returns: a pointer to the UArray2_T struct which represents the 2d
array
 *
 ************************/
extern T UArray2_new(int col, int row, int size);



/********** UArray2_width ********
 *
 * Returns the width (number of columns) of the given 2d array
 *
 * Parameters:
 *      T uarray2: the 2d array
 *
 * Returns: the width (number of columns) of the 2d array
 *
 ************************/
extern int UArray2_width(T uarray2);
```

```
/********** UArray2_height ********
 *
 * Returns the height (number of rows) of the given 2d array
 *
 * Parameters:
 *      T uarray2: the 2d array
 *
 * Returns: the height (number of rows) of the 2d array
 *
 ************************/
extern int UArray2_height(T uarray2);



/********** UArray2_size ********
 *
 * Returns the number of bytes each element occupies
 *
 * Parameters:
 *      T uarray2: the 2d array
 *
 * Returns: the number of bytes each element occupies
 *
 ************************/
extern int UArray2_size(T uarray2);



/********** UArray2_at ********
 *
 * Get the element at the given index
 *
 * Parameters:
 *      T uarray2: the 2d array
 *      int col:   the column of the intended result. Col should be less
than
 *                 the width of the array
 *      int row:   the row of the intended result. Row should be less than
the
 *                 height of the array
 *
 * Returns: a pointer to the element at the index. CRE if index out of
bounds
 *
 ************************/
extern void *UArray2_at(T uarray2, int col, int row);



/********** UArray2_map_col_major ********
 *
 * Calls apply on every element in the 2d array going column by column
 *
 * Parameters:
 *       T uarray2:    the 2d array
 *       void apply(): the function to be called on each element
 *       void *cl:     the closure variable
 *
 * Returns: none
 *
 ************************/
extern void UArray2_map_col_major(T uarray2,
                                  void apply(int i, int j, UArray2_T a,
                                             void *p1, void *p2),
                                  void *cl);



/********** UArray2_map_row_major ********
 *
 * Calls apply on every element in the 2d array going row by row
 *
 * Parameters:
 *       T uarray2:    the 2d array
 *       void apply(): the function to be called on each element
 *       void *cl:     the closure variable
 *
 * Returns: none
 *
 ************************/
```

```
extern void UArray2_map_row_major(T uarray2,
                                  void apply(int i, int j, UArray2_T a,
                                             void *p1, void *p2),
                                  void *cl);



/********** UArray2_free ********
 *
 * Deallocates and frees the memory in the array
 *
 * Parameters:
 *      T *uarray2: pointer to the 2d array. CRE if *uarray2 is null.
 *
 * Returns: none
 *
 ************************/
extern void UArray2_free(T *uarray2);

#endif
```

Invariants

- Each element in the array will occupy the same number of bytes (as indicated in UArray2_new
- Each column has the same amount of rows, and each row has the same amount of columns

```c
#ifndef BIT2_INCLUDED
#define BIT2_INCLUDED

typedef struct Bit2 *Bit2_T;

extern Bit2_T Bit2_new(int width, int height);
extern void Bit2_free(Bit2_T *a);

extern int Bit2_width (Bit2_T a);
extern int Bit2_height(Bit2_T a);

extern int Bit2_get(Bit2_T a, int i, int j);
extern int Bit2_put(Bit2_T a, int i, int j, int bit);

extern void Bit2_map_row_major(
        Bit2_T a,
        void apply(int i, int j, Bit2_T b, int value, void *cl),
        void *cl);

extern void Bit2_map_col_major(
        Bit2_T a,
        void apply(int i, int j, Bit2_T b, int value, void *cl),
        void *cl);

#endif
```