```
#ifndef UARRAY2_INCLUDED
#define UARRAY2_INCLUDED

#define T UArray2_T
typedef struct T *T;



extern T *UArray2_new(int row_dim, int col_dim, int elemSize);

/********** UArray2_new ********
 *
 * Purpose:
 *     - allocates space on the heap for a 2D array
 *       with the given row count, column count, and size of element that
 *      will be stored.
 *
 * Inputs:
 *     - int row_dim:
 *           the amount of rows to allocate space for
 *     - int col_dim:
 *           the amount of columns allocate space for
 *     - int elemSize:
 *           the size of a specific element type that will be stored in the
 *           array
 *
 * Return:
 *     - outputs a pointer to the UArray2 struct.
 *
 * Expects:
 *         TBD
 * Notes:
 *     - assert memory allocation.
 *     - allocates sizeof(UArray_T) on the heap.
 *
 ************************/
```

```
extern void UArray2_free(T *uArr_p);

/********** UArray2_free ********
 *
 * Purpose:
 *     - Frees the heap memory used during the
 *       initialization and allocation of the UArray2 instance.
 *
 * Inputs:
 *     - T *uArr_p:
 *           holds a pointer of the T instance
 *           to free.
 *
 * Return:
 *      - None (void)
 *
 * Expects:
 *     - Doesn't directly return anything, however
 *       indirectly returns NULL to the passed T *uArr_p.
 *
 * Notes:
 *     - assigns the passed T uArray to NULL.
 *
 ************************/
```

```
extern int UArray2_width(T uArr);

/********** UArray2_width ********
 *
 * Purpose:
 *     - returns the width (horizontal measure) of the 2D array.
 *
 * Inputs:
 *     - T uArr:
 *           holds a passed-by-value instance of a UArray.
 *
 * Return:
 *     - returns an integer which is the width of the
 *       UArray2.
 *
 * Expects:
 *         TBD
 * Notes:
 *         TBD
 ************************/
```

```
extern int UArray2_height(T uArr);

/********** UArray2_height ********
 *
 * Purpose:
 *     - returns the height (vertical measure) of the 2D array.
 *
 * Inputs:
 *     - T uArr:
 *           holds a passed-by-value instance of a UArray.
 *
 * Return:
 *     - returns an integer which is the height of the
 *       UArray2.
 *
 * Expects:
 *         TBD
 * Notes:
 *         TBD
 ************************/
```

```
extern int UArray2_size(T uArr);

/********** UArray2_size ********
 *
 * Purpose:
 *     - returns the size of an element capable of being stored in the uArr.
 *
 * Inputs:
 *     - T uArr:
 *           holds a passed-by-value instance of a UArray.
 *
 * Return:
 *     - returns an integer which is the size of an element
 *       inside the UArray2.
 *
 * Expects:
 *         TBD
 * Notes:
 *         TBD
 ************************/
```

```
extern void *UArray2_at(T uArr, int row, int col);

/********** UArray2_at ********
 *
 * Purpose:
 *     -  Access the element at the given [row][col] index
 * Inputs:
 *         - T uArr:
 *              holds a passed-by-value instance of a UArray
 *       - int row:
 *              the vertical index where the element is being accessed
 *       - int col:
 *              the vertical index where the element is being accessed
 * Return:
 *      - the element that exists at a given index, in the form of a void
 *        pointer to match the element type
 * Expects:
 *         TBD
 * Notes:
 *         TBD
 ************************/
```

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

/********** UArray2_map_row_major ********
 *
 * Purpose:
 *     - Iterates through an instance of a UArray2
 *       using row major iteration
 *
 * Inputs:
 *     - T uArr:
 *           holds the specific instance to a UArray2.
 *     - void apply:
 *           function which is applied to every
 *           element in the UArray2.
 *     - void *cl:
 *           pointer to a variable needed by the void apply
 *           function pointer.
 *
 * Return:
 *     - None.
 *
 * Expects:
 *     - Modifies the elements inside the UArray2 based on instructions
 *       provided in the void apply function.
 *
 * Notes:
 *     - an out of bounds reference will call a checked runtime error
 *
 ************************/
```

```
extern void UArray2_map_col_major(T uArr, void apply(int i, int j, UArray2_T a,
                    void *p1, void *p2), void *cl);

/********** UArray2_map_col_major ********
 *
 * Purpose:
 *     - Iterates through an instance of a UArray2
 *       using column major iteration
 *
 * Inputs:
 *     - T uArr:
 *           holds the specific instance to a UArray2.
 *     - void apply:
 *           function which is applied to every
 *           element in the UArray2.
 *     - void *cl:
 *           pointer to a variable needed by the void apply
 *           function pointer.
 *
 * Return:
 *     - None (void)
 *
 * Expects:
 *     - Modifies the elements inside the UArray2 based on instructions
 *       provided in the void apply
 *
 * Notes:
 *         TBD
 ************************/



#undef T
#endif
```

Bit2 functions

```
#ifndef BIT2_H
#define BIT2_H

#define T Bit2_T
typedef struct T *T;

extern T Bit2_new(int row_dim, int col_dim);

extern void Bit2_free(T *bArr_p);

extern int Bit2_width(T bArr);

extern int Bit2_height(T bArr);

extern void Bit2_put(T bArr, int row, int col, int marker);

extern void *Bit2_get(T bArr, int row, int col);

extern void Bit2_map_row_major(T bArr, void apply(int i, int j,
            T a, void *p1, void *p2), void *cl);

extern void Bit2_map_col_major(T bArr, void apply(int i, int j,
            T a, void *p1, void *p2), void *cl);


#undef T
#endif
```