Function contracts

UArray2

***

UArray2_T UArray_new(int height, int width, int size);

```
/********** UArray2_new ********
*
*       Allocates, initializes, and returns a new 2-dimensional  array of (height*width) elements
*       with bounds zero through (height − 1)(width - 1).
*
*       Parameters:
*               int width:      the width of the new uarray2
*               int height:     the height of the new uarray 2
*               int size:       the size of each element in the uarray2
*
*       Return:
*               the new uarray2
*
*       Expects:
*               height > 0 && width > 0 && size > 0
*
*       Notes:
*               Will CRE if height, width or size is <= 0
*               Will CRE if malloc fails.
*
************************/
```

void UArray2_free(UArray2_T *uarray2);

```
/********** UArray2_free ********
*
*       Deallocates and clears all elements in the uarray2.
*
*       Parameters:
*               UArray2_T *uarray2: The pointer to uarray2.
*
*       Return:  none
*
*       Expects:
*               Both the uarray2 and the pointer to uarray2 cannot be NULL
*
*       Notes:
```

```
*                   CRE if uarray2 == NULL or *uarray2 == NULL
*                   Sets the pointer to uarray2 to NULL
*
************************/
```

int UArray2_width(UArray2_T uarray2);

```
/********** UArray2_width ********
*
*         Checks for and returns the width of the uarray2.
*
*         Parameters:
*                   UArray2_T uarray2: the uarray2
*
*         Return: Int holding the width of the uarray2.
*
*         Expects:
*                   The uarray2 cannot be NULL
*
*         Notes:
*                   CRE if uarray2 == NULL
*
************************/
```

int UArray2_height(UArray2_T uarray2);

```
/********** UArray2_height ********
*
*         Checks for and returns the height of the uarray2.
*
*         Parameters:
*                   UArray2_T uarray2: the uarray2
*
*         Return: Int holding the height of the uarray2
*
*         Expects:
*                   The uarray2 cannot be NULL
*
*         Notes:
*                   CRE if uarray2 == NULL
*
************************/
```

int UArray2_size (UArray2_T uarray2);

/******** UArray2_size ***********

*

* Provides the size of each element in the uarray2 in bytes

*

* **Parameters:**

* UArray2_T uarray2: the uarray2

*

* **Return:** Int containing the size of an element in the uarray2 in bytes

*

* **Expects:**

* The uarray2 cannot be NULL

*

* **Notes:**

* CRE if uarray2 == NULL

*

**************************/**

void *UArray2_at(UArray2_T uarray2, int col, int row);

/******** UArray2_at ***********

*

* Finds the element at the (col, row) position of the uarray2

*

* **Parameters:**

* UArray2_T uarray2: the uarray2

* int col: the column number of the element in the uarray2

* int row: the row number of the element in the uarray2

*

* **Return:** A pointer to the desired element

*

* **Expects:**

* The uarray2 cannot be NULL

* Valid col, row coordinate that exists in uarray2

* **Notes:**

* CRE if uarray2 == NULL

* CRE if the position (col, row) is out of bounds

*

**************************/**

void UArray2_map_row_major (UArray2_T uarray2, void apply(int width, int height, UArray2_T uarray2, void *p1, void *p2), void *cl);

/********UArray2_map_row_major ***********

*

```
*        Applies an apply() function of choice to every value in the uarray2 in row-major order.
*
*        Parameters:
*                UArray2_T uarray2: the uarray2
*                void apply: the apply function, which applies to every element in the uarray2
*                void *cl: application-specific pointer
*
*        Return: none
*
*        Expects:
*                Valid apply function as well as a pre-initialized uarray2
*                NULL can be passed in instead of an application-specific pointer
*        Notes:
*                Apply function takes arguments: int width, int height,  UArray2_T uarray2,
*                                                        void *p1, void *p2
*                Arguments can be set as void if necessary
*                CRE if uarray2 == NULL
*
************************/
```

void UArray2_map_col_major (UArray2_T uarray2, void apply(int width, int height,
UArray2_T uarray2, void *p1, void *p2), void *cl);

```
/**********UArray2_map_col_major ********
*
*        Applies an apply() function of choice to every value in the uarray2 in column-major order.
*
*        Parameters:
*                UArray2_T uarray2: the uarray2
*                void apply: the apply function, which applies to every element in the uarray2
*                void *cl: application-specific pointer
*
*        Return: none
*
*        Expects:
*                Valid apply function as well as a pre-initialized uarray2
*                NULL can be passed in instead of an application-specific pointer
*        Notes:
*                Apply function takes arguments: int width, int height,  UArray2_T uarray2,
*                                                        void *p1, void *p2
*                Arguments can be set as void if necessary
*                CRE if uarray2 == NULL
*
************************/
```