```c
/*
*     uarray2.h
*     jadkin05, kdhaya01, 09/17/2024
*     iii
*
*     Contains all function headers for using UArray2_T.
*/

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef struct UArray2_T *UArray2_T;


/*
*   UArray2_new
*
*   Allocates space for and creates a 2D Uarray comprised of an outer
Uarray
*   of "inner" UArrays.
*
*   Parameters:
*       int cols:  The desired # of cols in the 2D Uarray (width)
*       int rows:  The desired # of rows in the 2D Uarray (height)
*
*   Returns:
*       UArray2_T:  The pointer to the 2D Uarray (start of the outer 1D
Uarray)
*
*   Expects:
*       All integer values being provided are positive
*
*/
UArray2_T UArray2_new(int cols, int rows, int ELEMENT_SIZE);

/*
*   UArray2_width
*
*   Gets the number of columns in the 2D Uarray (width)
```

```
*
*   Parameters:
*       UArray2_T array:    The UArray2 for which you want the width of
*
*   Returns:
*       int width:  The number of columns in the 2D Uarray (width)
*
*   Expects:
*       array is a UArray2 pointer
*
*/
int UArray2_width(UArray2_T array);

/*
*   UArray2_height
*
*   Gets the number of rows in the 2D Uarray (height)
*
*   Parameters:
*       UArray2_T array:    The UArray2 for which you want the height of
*
*   Returns:
*       int height:  The number of rowss in the 2D Uarray (height)
*
*   Expects:
*       array is a UArray2 pointer
*
*/
int UArray2_height(UArray2_T array);

/*
*   UArray2_size
*
*   Gets the element size of the boxes in the "inner" UArray
*
*   Parameters:
*       UArray2_T array:    The UArray2 for which you want the element
size of
*
*   Returns:
```

```c
*       int size:  The size in bytes of a box/element in the "inner"
UArray
*
*   Expects:
*       array is a Urray2 pointer
*
*/
int UArray2_size(UArray2_T array);

/*
*   UArray2_at
*
*   Sets the value of an element at the desired location with the UArray2
*
*   Parameters:
*       UArray2_T array:    The UArray2 you want to set a value for
*       int col, row:  The (col,row) location of the element's value to be
set
*
*   Returns:
*       void*:  void pointer
*
*   Expects:
*       array is a Urray2 pointer
*       Both integers are positive and within the dimensions of the array
*
*/
void* UArray2_at(UArray2_T array, int col, int row);

/*
*   UArray2_map_col_major
*
*   Maps the UArray2 doing so column by column (column major)
*
*   Parameters:
*       UArray2_T array:    The UArray2 you want to map
*       void (*func):  The function that checks for the corner of the
UArray2
*                      and prints out the current index
*       bool *ok:   Boolean stating if the Uarray2's dimensions are
correct
*
*   Returns:
*       void:   no return value
*       Prints out the every index of the UArray2 column by column
*
*   Expects:
*       array is a Urray2 pointer
*       The function being provided has matching parameter value types
*
*/
void UArray2_map_col_major(UArray2_T array, void(*func)(int, int,
UArray2_T, void *, void *), bool* OK);

/*
*   UArray2_map_row_major
*
*   Maps the UArray2 doing so row by row (row major)
*
*   Parameters:
*       UArray2_T array:    The UArray2 you want to map
*       void (*func):  The function that checks for the corner of the
UArray2
*                      and prints out the current index
*       bool *ok:   Boolean stating if the Uarray2's dimensions are
correct
*
*   Returns:
*       void:   no return value
*       Prints out the every index of the UArray2 row by row
*
*   Expects:
*       array is a Urray2 pointer
*       The function being provided has matching parameter value types
*
*/
void UArray2_map_row_major(UArray2_T array, void(*func)(int, int,
UArray2_T, void *, void *), bool* OK);
```

```c
/*
*   UArray2_free
*
*   Frees all memory that has been allocated for the UArray2
*
*   Parameters:
*       UArray2_T *array:    The address of the UArray2 you want to free
*
*   Returns:
*       void:   no return value
*
*   Expects:
*       array is a Urray2 pointer that has been allocated
*
*/
void UArray2_free(UArray2_T *array);
```

```c
/*
*     bit2.h
*     jadkin05, kdhaya01, 09/17/2024
*     iii
*
*     Contains all function headers for using Bit2_T.
*/


#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

typedef struct Bit2_T *Bit2_T;

/*
*   Bit2_new
*
*   Allocates space for and creates a bit
*
*   Parameters:
*       int cols:  The desired # of cols in the Bit2 (width)
*       int rows:  The desired # of rows in the Bit2 (height)
*
*   Returns:
*       Bit2_T:
*
*   Expects:
*
*
*/
Bit2_T Bit2_new(int cols, int rows);

int Bit2_width(Bit2_T array);

int Bit2_height(Bit2_T array);

int Bit2_put(Bit2_T array, int col, int row, int mark);

int Bit2_get(Bit2_T array, int col, int row);
```

```
void Bit2_map_col_major(Bit2_T array, void (*func)(int, int, Bit2_T, int,
void *), bool *OK);

void Bit2_map_row_major(Bit2_T array, void (*func)(int, int, Bit2_T, int,
void *), bool *OK);

void Bit2_free(Bit2_T *array);
```