Assignment #1: Dynamically Allocating Arrays

The Problem
Write several functions that dynamically allocate memory and use the memory to create, store and manipulate arrays.

Coding Guidelines
A style guide and other programming requirements will be posted to CourseLink.

The program must compile with the flags ***-Wall -ansi***

Project Structure
A header file will be provided with all function definitions (*arrayDefns.h*). Do not change the function prototypes – any alterations to the provided functions will result in a grade of 0.

You must hand in a zip file with 4 files:

arrayDefns.h – Header file with all function definitions (provided)
create2DArray.c – Function definitions for part 1
createStringArray.c – Function definitions for part 2
createArray.c – Function definitions for part 3

Mainline Testing Code
A mainline program will be provided on *CourseLink* to help you test your code. This will be provided closer to the due-date

Part 1: Create a Two-Dimensional Integer Array using an One-Dimensional Array

Write the following functions that dynamically allocate an one-dimensional array. The array is treated as a two-dimensional array.

You should only be using a single malloc() to create the array. The problem you need to solve is how to map the two indices that you would normally use with a two-dimensional array into the single index that you have with an one-dimensional array.

Functions

- int *create2DArray (int rows, int cols)
    - This function takes two parameters that are the number of rows and columns of the array you will be emulating. It returns a pointer to the data structure that you have dynamically allocated that has enough space to hold the array.

- void set2DElement (int *array, int row, int col, int value)
    - This function places a value in the array. The parameters are:

        - The *data structure* allocated using create2DArray()
        - The *row location* where the value will be stored
        - The *column location* where the value will be stored
        - The *value* to be placed in the array

- int get2DElement (int *array, int row, int col)
    - This function returns a value from the array. The parameters are:
        - The *data structure* allocated using create2DArray()
        - The *row location* where the value is stored
        - The *column location* where the value is stored

- void free2DArray (int *array)
    - This function frees the memory allocated by create2DArray(). The single parameter is the *data structure* allocated using create2DArray()

Part 2: Create an Array of Variable Length Strings

These functions create and access an array of strings.

Functions

- char **createStringArray (int number)
    - This function dynamically allocates an array of pointers that will point to the strings. It takes one parameter which is the size of the array of pointers (this corresponds to the number of strings that you wish it to hold).

- void setStringArray (char **array, int index, char *string)
    - This function stores a string in the array. The string passed into the function must have memory already allocated for it and must already be copied into that space. The parameters are:
        - The *array of strings* allocated using createStringArray()
        - The *location* in the array where the string will be stored
        - The *string* to store in the array

- char *getStringArray (char **array, int index)
    - This function returns a string from the array. The parameters are:
        - The *array of strings* allocated using createStringArray()
        - The *location* in the array where the string is stored

- void freeStringArray (char **array, int number)
    - This function frees the memory allocated using createStringArray() **and** any strings stored in the array. The parameters are:
        - The *array of strings* allocated using createStringArray()
        - The *size* of the array

Part 3: Create a Two-Dimensional Array

These functions create and free a dynamically allocated two-dimensional array. The array should function the same as a statically allocated array once it has been created.

Functions

- int **createArray (int rows, int cols)
    - The parameters to this function are the number of rows and columns in the array. The return value is the array that you have dynamically allocated.
- void freeArray (int **array, int rows, int cols)
    - The parameters to this function are:
        - The *array* created by createArray()
        - The *number of rows* in the array
        - The *number of cols* in the array