LAB #9 – Arrays vs. Structs

In this lab, you will create a two dimensional array of structs that contain the multiplication and division table of the rows and columns, starting at 1 instead of zero. This prevents us from causing a divide by zero error in the division table. The program needs to read the number of rows and columns from the user as command line arguments. You do not have to do error checking, but you will have to convert the string to a number.

    rows=atoi(argv[1]);   cols=atoi(argv[2]);

For example, if you run your program with these command line arguments, ./prog 5 5, then your program should create a 5 by 5 matrix of structs and assign the multiplication table to the mult variable in the struct and the division of the indices to the div variable in the struct. The mult variable is an integer and the div variable needs to be a float (or double).

    struct mult_div_values {
        int mult;
        float div;
    };

Your program needs to be well modularized with functions, including main, with 15 or less lines of code. This means you will have a function that creates the matrix of structs given the m x n dimensions, mult_div_values** create_table(int m, int n). You need to have functions that set the multiplication values and division values, void set_mult_values(mult_div_values **table, int m, int n) and void set_div_values(mult_div_values **table, int m, int n). Then, call functions to print the tables.

Example: ./prog 5 5

Multiplication Table:
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25

Division Table:
1.00 0.50 0.33 0.25 0.20
2.00 1.00 0.67 0.50 0.40
3.00 1.50 1.00 0.75 0.60
4.00 2.00 1.33 1.00 0.80
5.00 2.50 1.67 1.25 1.00

Show your program to a lab TA for 5 points of this lab.
More Structs:
In this lab, you will practice using structs in your program. We use structs as containers to hold information of mixed types, and many times this includes records of information in the real world, such as information about movies for renting. If you had to write a program for a redbox or Netflix, then you would have thousands of records of information about movies. How do you carry around this information in your program? Do you make an array for each piece of information? For example:

```c
#define NUM_MOVIES 100000
int main() {
    string title[NUM_MOVIES];
    int copies[NUM_MOVIES];
    string rating[NUM_MOVIES];
    string description[NUM_MOVIES];
    string genre[NUM_MOVIES];
    ...
```

Think about keeping up with what movie is in which slot of the arrays. For instance, all the information in all the elements must match, and if you sort the movies by title/genre or add/delete movies from your inventory, then you always have to keep track of what you do and apply the same action to all other arrays to keep them aligned, right? Wouldn’t this be easier to make a movie record of this information in a struct, and make one array of movie record structs? Your job is to do just this. Make a movie_record struct, and then make an array of movie_record structs.

After you setup your array of movie records, then write a function, `set_movies()`, that allows you to set the information for each movie, i.e. the title, rating, description, copies, and genre. You only have to do this for 5 movies, i.e. `#define NUM_MOVIES 5`.

Now, write another function, `which_movies_to_view()`, that asks the user which movies he/she wants to view. You can ask the user if he/she want to see all movies or by genre, and write a function, `get_movies()`, to display the titles of movies depending on the user’s choice. Hint: This might be a good time to use function overloading.

Lastly, let the user choose to see more information about a movie or rent a movie. Write a function, `get_movie_info()`, that displays all the information about a specific movie for a user, and write another function, `rent_movie()`, that rents the movie to the user by decrementing the number of copies for a specific movie.

Show your program to a lab TA for the other 5 points of this lab.