Lab 1

Due No Due Date **Points** 15 **Submitting** a file upload

Introduction & Review

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 5 points for lab work completed outside of lab time, but you must finish before the next lab. For extenuating circumstances, contact your lab TAs and the instructor.

Structs, Pointers, Arrays

In this lab, you will create a dynamic two dimensional array of mult_div_values structs (defined below). The two dimensional array will be used to store the rows and columns of a multiplication and division table. Note that the table will start at 1 instead of zero, to prevent causing a divide-by-zero error in the division table!

```
struct mult_div_values {
    int mult;
    float div;
};
```

The program needs to read the size of the matrix from the user as a command line argument. You should check that the user actually supplied a number before converting the input string to an integer. Continue to prompt the user for correct values if the number isn't a valid non-zero integer. At the end of the program, prompt the user if they want to see this information for a different size matrix.

```
// One approach to handling the command line arguments...
n=atoi(argv[1]);
// You'll need to put #include <stdlib.h> at the beginning of your program for this to work.
// Now check that it is a valid non-zero integer
// ...
```

For example, if you run your program with these command line arguments

```
./prog 5
```

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).

Your program needs to be well modularized with functions, including main, with 10 or less lines of code. This means you will have a function that checks if the dimension is a valid, non-zero integer:

```
bool is_valid_dimension(char *n)
```

and another function that creates the matrix of structs given the n dimension:

```
mult_div_values** create_table(int n)
```

In addition, you need to have functions that set the multiplication and division values, as well as delete your matrix from the heap:

```
void set_mult_values(mult_div_values **table, int n)
void set_div_values(mult_div_values **table, int n)
void delete_table(mult_div_values **table, int n)
```

Then create functions to print the tables. After your code is functioning, test it using the following approach. Once you are done, show your work to a TA for grading.

Example Run:

```
./prog t
You did not input a valid size.
Please enter an integer greater than 0 for a matrix: 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
Would you like to see a different size matrix (0-no, 1-yes)?
0
```

Remember, you will not receive lab credit if you do not get checked off by the TA before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of knowing if you were there or not!

Lab 1

Criteria	Ratings		Pts
User interface Checks if user has entered valid argument, prompts if not. Displays multiplication and division tables. Asks user if they'd like to go again, and exits if not.	**5 pts Full Marks**	**0 pts No Marks**	5 pts
Modularized code All functions are 10 lines of code or under.	**5 pts Full Marks**	**0 pts No Marks**	5 pts
Struct creation and deletion Properly creates and deletes tables and structs.	**5 pts Full Marks**	**0 pts No Marks**	5 pts
			Total Points: 15