Project\_1\_Movie\_Table

Start Assignment

* **Due** Feb 10 by 11:59pm
* **Points** 50
* **Submitting** a file upload
* **File Types** c
* **Available** Jan 27 at 12am - Feb 10 at 11:59pm 15 days

Overview

In this project, we will be reading a csv file with information about popular movies and printing that data in a nicely formatted table.

Learning Goals

Practice working with basic C programming, strings or arrays of characters, and formatted printing.

Corrections and Additions

* Do not include ctype.h - the library that includes toupper() and tolower()

Project Specifications Special Rules

* You may not use string.h. The goal is to gain practice working with C-style arrays of primitives.
* You may not use ctype.h. Another goal is to gain an understanding of how C deals with the low-level details of storing and using characters. It treats them all as numbers so con version from upper case to lower case can be done with arithmetic.

Files

[movies1.csv](https://canvas.wisc.edu/courses/280031/files/24319671?wrap=1) Table1.txt  :: input and sample output

movies2.csv Table2.txt :: inut and sample output

[driver.c](https://canvas.wisc.edu/courses/280031/files/24319717?wrap=1): don’t change this. This file calls the functions in student \_ functions that you will write

[student\_functions.h](https://canvas.wisc.edu/courses/280031/files/24319673?wrap=1): header file so your functions can be compiled with the driver

[student\_functions.c](https://canvas.wisc.edu/courses/280031/files/24319672?wrap=1): Complete these functions. Do all of your work here.

Movies.csv

These comma-separated value files contain data about several movies – one movie per row. The data include the following fields: title, year, runtime, director, rating, and revenue.

Functions

Descriptions of the functions required are found in the student\_functions files

You may write any additional helper functions you wish in student\_functions.c remember to include a prototype before using the function.

Compiling

Compile and run your code with

gcc -g -o Print\_Table student\_functions.c driver.c -Wall  
./Print\_Table

Testing your code

Capture the output into a file with output redirection. The compare to the solution with diff. Diff produces no output if the tables match exactly. Otherwise, it tells you which lines are different.

./Print\_Table > my\_output1.txt  
diff my\_output1.txt table1.txt

The test cases do not address all possible inputs.

Turn in

Upload your student\_functions.c file to Canvas

Style

We are not penalizing for compiler warnings on the first assignment, but it is good practice to address all of them.

Please see the [style guide](https://canvas.wisc.edu/courses/280031/files/24252423?wrap=1)

Student\_functions.c  
/\* This function takes a string as input and removes   
 \* leading and trailing whitespace including spaces  
 \* tabs and newlines. It also removes multiple internal  
 \* spaces in a row. Arrays are passed by reference.  
 \*/  
void Clean\_Whitespace(char str[]) {  
 // do your work here  
 return;  
}

/\* This function takes a string and makes the first  
 \* letter of every word upper case and the rest of the  
 \* letters lower case  
 \*/   
void Fix\_Case(char str[]) {  
 // do your work here  
 return;  
}

/\* this function takes a string and returns the   
 \* integer equivalent  
 \*/  
int String\_To\_Year(char str[]) {  
 // do your work here  
 return 0;  
}

/\* this function takes the name of a   
 \* director as a string and removes all but  
 \* the last name. Example:  
 \* "Bucky Badger" -> "Badger"  
 \*/  
void Director\_Last\_Name(char str[]) {  
 // do your work here  
 return;  
}

/\* this function takes the a string and returns  
 \* the floating point equivalent  
 \*/  
float String\_To\_Rating(char str[]) {  
 // do your work here  
 return 0.0;  
}

/\* this function takes a string representing  
 \* the revenue of a movie and returns the decimal  
 \* equivlaent. The suffix M or m represents millions,  
 \* K or k represents thousands.  
\* example: "123M" -> 123000000   
\*/  
long long String\_To\_Dollars(char str[]) {

// do your work here  
 return 0;  
}

/\* This function takes the array of strings representing   
 \* the csv movie data and divides it into the individual  
 \* components for each movie.  
 \* Use the above helper functions.  
 \*/  
void Split(char csv[10][1024], int num\_movies, char titles[10][1024], int   
years[10], char directors[10][1024], float ratings[10], long long dollars[10]) {  
 // do your work here  
 return;  
}

/\* This function prints a well formatted table of  
 \* the movie data   
 \* Row 1: Header - use name and field width as below  
 \* **Column 1:** Id, field width = 3, left justified  
 \* **Column 2:** Title, field width = length of longest movie + 2 or 7 whichever is larger, left justified, first letter of each word upper case, remaining letters lower case, one space between words  
 \* **Column 3:** Year, field width = 6, left justified  
 \* **Column 4:** Director, field width = length of longest director last name + 2 or 10 (whichever is longer), left justified, only last name, first letter upper case, remaining letters lower case  
 \* **column 5:** Rating, field width = 5, precision 1 decimal place (e.g. 8.9, or 10.0), right justified  
 \* **column 6:** Revenue, field width = 11, right justified  
 \*/  
void Print\_Table(int num\_movies, char titles[10][1024], int years[10], char   
directors[10][1024], float ratings[10], long long dollars[10]) {  
 // do your work here  
 return;

}