Assignment Summary

For this assignment you will write a program that loads data from a data file into arrays of structs. The program will do some calculations using the data, print out information, and exit. The program must take the name of the data file using argv and argc.

Details

The data file will contain fictional weather data for two cities: Saskatoon and Guelph. Each row of data contains the city name, the date, the high temperature, the amount of precipitation. For this assignment dates are represented as the number of days since Jan 1 (i.e. Jan 31 is day 31, Feb 5 is day 36, etc). A function to parse one line of this data and copy the data into a struct supplied as a parameter to the function has been provided at the bottom of this assignment description.

Your program must do the following:

- Take the name of the data file using argv/argc. Do not prompt the user for a data file unless there is an error opening the file.
- Open the data file for reading. There will be no more than 200 lines of data in the data file. There will be a newline at the end of the data file.
- Create two arrays of struct Weather (struct definition given below). One array is for data from Saskatoon, one array is for data from Guelph.
- Read the data line by line. Decide whether the line goes into the Guelph array or the Saskatoon array.
- use the parser provided to decode the line into a struct in the array
- once all of the data file has been read, calculate the average temperature for each city given the data.
- calculate the total precipitation for each city
- print the results in the required format

Required Output

The output for this program must be exactly as shown. If you change the location of anything, even a space or a blank line, you will lose marks.
Start the output with one blank line:

```

******
Guelph
******
AVG TEMP: 12.00
TOTAL PRECIP: 8.00
```

Leave a single blank line then print each element of the Guelph array, one per line, as specified in the description of the print function. Leave a single blank line then begin the section for Saskatoon.

```
*********
Saskatoon
*********
AVG TEMP: 3.00
TOTAL PRECIP: 2.00
```

Leave a single blank line then print each element of the Saskatoon array, one per line, as specified in the description of the print function. Do not print a blank line after the Saskatoon list.

Required Functions

1. A function to print a single struct. It must output the data of the struct exactly as specified: DAY: daynum (TEMP: temp)[precip MM]
   a. daynum, temp and precip are the values. All other text should be reproduced exactly as shown, including the spaces. Numbers should be printed to two decimal points.
   b. The function should not print a \n after the text.
   c. For example, suppose the data were: City:Guelph, Day:133, Temp:25.01, Precipitation:0.0, the print function would print a single line of text that looked like this: DAY: 133 (TEMP: 25.00)[0.00 MM]
2. A function that returns a file pointer after open the file and checking for null file pointer and prompting the user to re-enter the filename if necessary.
3. A function that returns a double to calculate the average temperature given an array as a parameter.
4. A function that returns a double to calculate the total precipitation given an array as a parameter.

The required functions may have any suitable name and parameter list

Struct definition (required- do not change the struct)

```
struct Weather
{
        char location; //will be S or G
        int daynum;
        double temp;
        double precip;

};
```

Parser

You may use the parser given below or you may write your own. If you use this one you must add comments to the code that show that you understand what the parser is doing and you must include a comment indicated that you used the parser provided with the assignment description.

```
void parseLine(char toParse[], struct Weather * toLoad)
{
   char * theToken;
   theToken = strtok(toParse, ",");
   toLoad->location = theToken[0];
   theToken = strtok(NULL, ",");
   toLoad->daynum = atoi(theToken);
   theToken = strtok(NULL, ",");
   if(theToken != NULL)
   {
      toLoad->temp = atof(theToken);
   }
   else
   {
      toLoad->temp = -400;
   }
   theToken = strtok(NULL, ",");
   if(theToken != NULL)
   {
      toLoad->precip = atof(theToken);
   }
   else
   {
      toLoad->precip = -1.0;
   }
}
```

Submission requirements and marks

- Submit only the c file for this assignment. Do not submit any data files or executables
- Call your c file weather.c
- You must follow coding conventions consistent with the course website or textbook
- You must use function prototypes and include comments describing each function as shown in the course coding conventions
- Marks will be allocated as follows:
    - Coding conventions, indentation, variable names, style 20%
    - Correct use of struct and creation of functions for managing struct 15%
    - Correct use of input file and reading file from command line 15%
    - Correct output 50%