Computer Science 327
Homework 2

This assignment is part of a larger project that will analyze weather data from the weather stations across the world. For this class, we will use weather data from weather stations in or near Ames, Iowa. In this first part of the assignment you will write a package of functions that reads and writes weather data in a specific format.

Weather data is stored in a variety of formats. For this project, the data is stored as "comma separated values" and the first row contains header data that indicates what data for each column contains. These data files are in plain ASCII format so it is easy to look at them to get an idea of what the data looks like. The exact format as well as what the data is described in the DataFormat file. Note that this is a 350 page document, but you will only need to look at a few pages that are relevant to the types of data stored in AmesWeatherData. (See the header row in this file.)

The AmesWeatherData file is an ASCII (plain character) text file with data values separated by commas. Each line of data contains entries for one row of data. Remember that the first row contains headers (not the data) that tell what the data in that column is. Thus, the number of items in the header row can tell you how many entries to read for each subsequent row. Your goal is to create a set of functions that will read and parse this file.

You are to store all functions related to reading the weather data in a file named weatherio.c and make appropriate use of the header file named weatherio.h. You must implement the following functions exactly using the prototypes specified below.

```
int validate_format( char * );
```

This function returns 0 if the file name specified parameter contains a valid formatted weather data file. The function returns the first line number that an error occurred. At a minimum you must validate the following:

Not enough data in row
Invalid data format
Invalid file or file name

```
int header_columns( char *);
```

This function returns the number of data columns (header columns) in the file name specified in the first parameter.

```
int read_header( char *, char ** );
```

This function returns the number of data columns in the file, and in addition, returns the c-strings for those headers in an array that is passed as the second parameter. You may safely assume that no header string is greater than 64 characters.

```
int read_row( FILE *, char **, void **);
```

This functions returns a row of data in an open file and assumes that the file pointer is positioned at a row passed the header file. The first parameter is a FILE * type to an open file. The second parameter is the array of c-strings that are the headers, as returned by the read_header function. The last parameter is an array of pointers where each pointer points to the appropriate type for the item read in the column as defined by the header information. Note that you will need to dynamically allocate memory for this to work. The single return value returns 0 if the row was read correctly, and otherwise returns an integer that indicates the type of error. You may diecide on the error codes, but they should be well documented and placed in the header (.h) file.

Finally you are to write a main method and place it in the file named testformat.c The main program uses a filename from the first command line argument and then verifies that the file is a valid formatted file. The main program outputs the string to stdout “valid” or “not valid” depending if the file is a valid formatted weather data file. Your executable must be called “testformat”

You must create a makefile that by default will compile and create the executable described above. If you have other utility functions please put them in either the weatherio.c file if appropriate, or a file named util.c.

Please feel free to post smaller test weather data files with and without errors on blackboard. They will help everyone with debugging their code.