CIS*1500 – Assignment 4

Due: Sunday March 20 – 11:59pm

Write a complete C program for the following problem and submit to the dropbox for Assignment 4. **Make sure you submit the .c file, not the executable.** Your programs must compile successfully with the command: **gcc -Wall -std=c99 your_file.c**.

The CSV (comma separated values) file format is a very common way to export data from spreadsheet and web-based applications. In these text files, each line in the file represents a row of data, where within each line, the different data fields are separated by a comma. For this problem you will be given a CSV file **grades.txt** containing assignment grades for every student in a given class. As an example, a CSV file for a class with 4 students and 3 assignments may look like:

```
8.5, 10.5, 90.5
50, 99, 97
88, 88, 100
88.5, 99, 0
```

Your task is to write 3 **functions** to compute and return:

- ▷ the average grade
- ▷ the maximum grade
- ▷ the number of students who failed ($< 50\%$) (use 49.9 in your comparison!)

for a given assignment (passed as one of the parameters). You may assume:

- every assignment is out of 100,
- each mark in the input file is rounded to the nearest half mark,
- there are between 1 and 100 assignments in the course,
- there are between 1 and 1000 students in the class,
- the file is formatted correctly.

To simplify reading the input file, your executable **must** take two command line parameters specifying the number of students and the number of assignments in the class. If the executable is **a.out**, and the input file has 4 students and 3 assignments as above, you would call your program as: **./a.out 4 3**. If an incorrect number of command line arguments is given, your program should output:

usage: ./a.out num_students num_assignments

NOTE: If the command line arguments do not match the input file, then your program is not expected to work, and appropriate error messaging is not required.

To store the grades, declare a **global variable** as follows: **float grades[1001][101];** (Using 1001 instead of 1000 allows you to start your indices at 1 if you wish). Since this 2-dimensional array is **global**, you do not have to pass it into your functions. This should be your **only** global variable. As an example, you may want to store the grade of the 4th assignment of the 9th student in `grades[9][4]`.

Finally, you must output grades data back to the screen in CSV format to verify you read the file correctly. Given the earlier CSV data, your output should be formatted as follows. Be mindful of the nice **alignment** of the values. The average should be **rounded** to one decimal place.

```
================================
Student grades from the input file
================================
  8.5,  10.5,  90.5
 50.0,  99.0,  97.0
 88.0,  88.0, 100.0
 88.5,  99.0,   0.0

==============
Ass # 1 stats
==============
  avg =  58.8
  max =  88.5
 fail =     1

==============
Ass # 2 stats
==============
  avg =  74.1
  max =  99.0
 fail =     1

==============
Ass # 3 stats
==============
  avg =  71.9
  max = 100.0
 fail =     1
```

Extra programming fun: Write a function to compute the **median** and **standard deviation** for each assignment.