Assignment 1 - Statistics

This assignment asks you to write a Bash script to compute statistics. The purpose is to get you familiar with the Unix shell, shell programming, Unix utilities, standard input, output, and error, pipelines, process ids, exit values, and signals.

What you're going to submit is your script, called statistics.

Overview

NOTE: For this assignment, make sure that you are using Bash as your shell!

In this assignment you will write a Bash script to calculate averages and medians from an input file of numbers. The input file will have whole number values separated by tabs, and each line of this file will have the same number of values. For example, each row might be the scores of a student on assignments. Your script should be able to calculate the average and median across the rows or down the columns.

You will probably need commands like these, so please read up on them: sh, read, expr, cut, head, tail, wc, and sort.

Your script will be called statistics. The general format of the statistics command is

```
statistics {-rows|-cols} [input_file]
```

Note that when parameters are in curly braces separated by a vertical bar, it means you should choose one of those parameters; here for example, you must choose either -rows or -cols. The option -rows calculates the average and median across the rows; the option -cols calculates the average and median down the columns. When parameters are in square braces it means they are optional; you can include them or not, as you choose. If you specify an input_file the data is read from that file; otherwise, it is read from standard input.

Here is a sample run of what your script might return, using the test_file, note that if you open this using notepad in Windows, the newline characters may not display as newlines. You need to open it with an editor that support this feature like Notepad++, and even there you have to select the feature in settings. You may be able to see them on your UNIX account if you cat it out (use the cat command).

```
% cat test_file
```

```
 1       1       1       1       1
 9       3       4       5       5
 6       7       8       9       7
 3       6       8       9       1
 3       4       2       1       4
 6       4       4       7       7
```

```
% statistics -rows test_file
 Average Median
 1       1
 5       5
 7       7
 5       6
 3       3
 6       6
```

```
% cat test_file | statistics –c
 Averages:
 5       4       5       5       4
 Medians:
 6       4       4       7       5
```

```
% echo $?
 0
```

```
% statistics
 Usage: statistics {-rows|-cols} [file]
```

```
% statistics -r test_file nya-nya-nya
 Usage: statistics {-rows|-cols} [file]
```

```
% statistics -both test_file
 Usage: statistics {-rows|-cols} [file]
```

```
% chmod -r test_file
```

```
% statistics -columns test_file
 statistics: cannot read test_file
```

```
% statistics -columns no_such_file
 statistics: cannot read no_such_file
```

```
% echo $?
 1
```

Specifications

1. You must check for the right number and format of arguments to statistics. You should allow users to abbreviate -rows and -cols; any word beginning with a lowercase r is taken to be rows and any word beginning with a lowercase c is taken to be cols. So, for example, you would get averages and medians across the rows with -r, -rowwise and -rumplestiltskin, but not -Rows. If the command has too many or two few arguments or if the arguments of the wrong format you should output an error message to standard error. You should also output an error message to standard error if the input file is specified, but it is not readable.

2. You should output the statistics to standard output in the format shown above. Be sure all error messages are sent to standard error and the statistics are sent to standard output. If there is any error, the exit value should be 1; if the statistics program runs successfully the exit value should be 0.

3. Your statistics program should be able to handle files with any reasonable number of rows or columns. You can assume that each row will be less than 1000 bytes long (because Unix utilities assume that input lines will not be too long), but don't make any assumptions about the number of rows. Think about where in your program the size of the input file matters. You can assume that all rows will have the same number of values; you do not have to do any error checking on this.

4. You will probably need to use temporary files. For this assignment, the temporary files should be put in the current working directory. A more standard place for temporary files is in /tmp, but don't do that for this assignment; it makes grading easier if they are in the current directory. Be sure the temporary file uses the process id as part of its name, so that there will not be any conflicts if the statistics program is running more than once. Be sure you remove any temporary files when your statistics program is done. You should also use the trap command to catch interrupt, hangup, and terminate signals to remove the temporary files if the statistics program is terminated unexpectedly.

5. All values and results are and must be whole numbers. You may use the expr command to do your calculations, or any other bash shell scripting method, but you may not do the calculations by dropping into another language, like awk, perl, python, or any other language. You may certainly use these other languages for all other parts of the assignment. Note that expr only works with whole numbers. When you calculate the average, you should round to the nearest whole number, where half values round up, i.e., 7.5 rounds up to 8. This is the most common form of rounding. You can learn more about rounding methods here (see Half Round Up):

http://www.mathsisfun.com/numbers/rounding-methods.html

6. To calculate the median, sort the values and take the middle value. For example, the median of 97, 90, and 83 is 90. The median of 97, 90, 83, and 54 is still 90; when there are an even number of values, choose the larger of the two middle values.

7. Your script, statistics, must be entirely contained in that file. Do not split this assignment into multiple files or programs.

8. To make it easy to see how you're doing, the actual grading script is included in the assignment's folder. Compare your output to a perfect solution, *p1cleantestscript* file that shows what you should get if everything is working correctly

9. The p1gradingscript itself is a good resource for seeing how some of the more complex shell scripting commands work, too.

Hints

- One problem that will be especially challenging is to read in the values from a specified file. The read command is exactly what you need (see the man page for read). However, read is meant to read from standard input and the input file is not necessarily standard input to the statistics command. You will have to figure out how to get read to read from a file. The man page for sh has the information you need to figure this out.
- Another problem will be calculating the median. There is a straightforward pipelined command that will do this, using cut, sort, head, and tail. Maybe you can figure out other ways, too. Experiment!
- The expr command and the shell can have conflicts over special characters. If you try expr 5 * (4 + 2), the shell will think * is a filename wild card and the parentheses mean command grouping. You have to use backslashes, like this: expr 5 * (4 + 2)

Grading

Grading will be based on successfully passing the grading script, AND on your style, readability, and commenting. Comment well, often, and verbosely. We mainly want to see that you are telling us WHY you are doing things, but also to tell us WHAT you are doing if it's needed. Read the document about proper documentation for this.