The goal of this homework is to explore basic C and MIPS programming skills needed for the project. It also explores File I/O and processing command-line arguments.

HW2-1 In this part, do two tasks:

(1) Write a C program that computes the median of a 100 element integer array. The integers in the array will range from -999 to 999. The array may be modified by your program, but the result computed by your program should be the median of the original array. The result should be actual, not a statistical approximation. The result is an integer.

Your program works with a previously defined 100 element vector stored in a file. A shell program is provided to help you get started. The included `printf` statement should be used to print your result so it can be properly graded. Its output format should not be modified.

(2) Modify your program to allow it to accept an additional command-line argument which is the name of an output file. Your program should write the median result to a file of that name using the same print format string used in the `printf` statement in (1).

While there are many median code examples available (e.g., on the web), you should design, implement, and test your own code. Otherwise you won't learn the things you need to know for later assignments and exams. **Any submitted project containing code not fully created and debugged by the student constitutes academic misconduct.**

The shell program `HW2-1-shell.c` includes a reader function `Load_Mem()` that loads the values from a text file. You should use `gcc` under Ubuntu to develop your program. Sample value files (`test-12.txt`, `test41.txt`, `test55.txt`, `test62.txt`, `test118.txt`) are provided on the project website. You should compile and run your program using the Linux command lines:

```
> gcc HW2-1.c –g –Wall –o HW2-1
> ./HW2-1 test215.txt median215.txt
```

In this example, the input test case is the value file `test215.txt` and `median215.txt` is the name of the file to which the program should write the median result.

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named `HW2-1.c`.
2. You name and the date should be included in the header comment.
3. The starting *shell* program should not be modified except for the replacement of the comment `/* your program goes here */` and the addition and/or initialization of declared local variables. It is especially important not to remove or modify the print statement since that will be used in the automatic grading process.
4. Your solution must be properly uploaded to the submission site before the scheduled due date, 5**:00pm on Friday, 1 February 2013**.

HW2-2: In this part, you will write an assembly program `Median` that computes the median of 100 integers stored in memory. You will call a software interrupt that will store 100 randomly generated numbers in static memory you have reserved. A shell program `HW2-2-shell.asm` is provided to get you started.

As in part HW2-1, there may be code available from other sources (e.g., on the web, from friends/enemies, etc.); don't use it! You should figure this out on your own.

Library Routines: There is one library routine (accessible via the `swi` instruction).

SWI 542: Random: This routine initializes memory beginning at the specified base address (e.g., `Array`) with 100 random integers. The integers are in the range -999 to 999. INPUTS: $1 should contain the base address of the 100 words already allocated in memory. OUTPUTS: none.

During testing, from Misasim you can dump the memory array to a text file and run your HW2-1 program on it to see if both your C and ASM programs are getting consistent results. (This is how the value files provided in HW2-1, such as `test41.txt`, were created.)

In order for your solution to be properly received and graded, there are a few requirements.

1. The file must be named `HW2-2.asm`.
2. Your name and the date should be included in the beginning of the file.
3. Your program must produce and store the median value in $2 when it returns.
4. Your program must return to the operating system via the **`jr`** instruction. *Programs that include infinite loops or produce simulator warnings or errors will receive zero credit*.

Your solution must be properly uploaded to the submission site before the scheduled due date, 5**:00pm on Friday, 1 February 2013**.

Project Submission Instructions:

To submit your project, you will upload the answer to each part of the assignment as a separate file. When you are ready to submit a file as your answer to a part of the assignment, use your web browser to go to the web site: http://www.ece.gatech.edu/~linda/2035/projects/index.html and click on FILE UPLOAD, provide your user id and the file to upload.

Double check that the name of the file uploaded is the one you intended to submit.

If you submit a file as the answer to part of the project and later you would like to submit an improved answer, you may submit the more recent version of the file. Only the most recent one will be graded. However, versions submitted after the due date will not be graded.