Name:

|  |  |  |
| --- | --- | --- |
| Problem | Points | Score |
| 0 | 50 | Simply show up… |
| 1 | 60 |  |
| 2 | 70 |  |
| 3 | 80 |  |
| 4 | 90 |  |
| 5 | 100 |  |
| Total | 100 |  |

Notes:

1. For this exam you are allowed to open a terminal window on your computer, you are allowed to web surf with Google, but you cannot use online chat or other interactive services.
2. Your code and results should be placed in directories p01, p02, …, p06.
3. You must work the problems in order – you cannot skip a problem. For example, you cannot complete p03 until you have finished p02.

As usual, place your work here with the proper permissions:

/data/courses/ece\_1111/current/exams/exam\_04/lastname\_firstname

You must use Python. Your programs must run using a name like “p01” or “p02”. If I can’t run your code typing such a command name, you will fail the exam. In each problem directory, I will expect to run your program by typing “p01 <arguments>”. Any deviation from this interface will result in a score of 50.

This exam is structured in a tiered manner. Start with problem 1. When done, copy your code to the next problem and continue editing it. Only turn in code that is completely working. I will grade the highest level you submit. If it doesn’t work, you will get a 50 for this exam (we call this the “you didn’t debug your code and wasted my time” penalty). Therefore, what you submit must work and meet the stated requirements.

If you write good clean code for problem no. 1, you can reuse that code for the remaining problems and quickly solve those problems.

**Problem No. 1**: Files in engineering often have a header, some metadata, and then actual signal data. The header usually occupies a fixed size, for example 1024 bytes, and describes the following signal data. In this problem, you will write a Python program to generate such a file.

Your header will be simple. The first four bytes will contain the size of the header as a 32-bit unsigned integer. Within this header, the first 8 bytes will contain “ECE 1111”. The next 4 bytes will contain the size of the signal data in “samples”. Samples are assumed to be 16-bit signed integers. The remaining bytes in the header will contain a NULL character. The entire header, including the first field containing the size of the header, fits within the number of bytes allocated to store the header.

For example, in the above example, if the header size was $100 bytes$, the non-zero header data will contain $4+8+4=16 bytes$ of information, leaving $84$ bytes empty.

The interface should be this:

p01 <header\_size> <signal size> <filename>

For example, this command line:

p01 1024 100 p01.raw

will create a file that has $1024$ bytes of a header, and $100$ signal samples stored as signed 16-bit integers. The total size of the file will be $1024 + 100 \* 2 = 1,224 bytes$.

For this problem, fill the signal data with a value of “$27$”, so we can debug your program. Your generator program must work for any header size and signal data size.

**Problem No. 2:** Write a program that reads the file generated in problem no. 1, sets the signal values to random numbers between $[-range, range]$, and writes a new file with the revised signal values:

p02 <range> <input file> <output file>

The output file should contain random signal values in the range $[-range,range]$ instead of “$27$”.

**Problem No. 3:** Write a program that reads your file and computes the standard deviation (stdev) of the signal. You MUST code this yourself – you cannot use a library function to compute the standard deviation. The standard deviation is defined as:

 $$σ=\sqrt{(\frac{1}{N})\sum\_{i=0}^{N-1}(x\left[i\right]-mean)\*(x\left[i\right]-mean)}$$

The program should print these values to stdout:

p03 <filename>

the mean = xxxxx.xxx

the standard deviation = xxxxx.xxxx

The mean is simply the arithmetic average. The mean of your noise signal in problem no. 2 should be close to zero. The standard deviation should be related to “$range$”. What is that relation?

Put your answer in a file named AAREADME.txt in the same directory.

**Problem No. 4:** Write a program that reads a file of the format above, sorts the samples into ascending order, and writes a new file with the sorted values:

p04 <input file> <output file>

Use the od command to check your results.

Run the file generated in problem no. 2 through this code and determine if the mean or standard deviation changed from what was computing in problem no. 3. Was this expected?

Put your answer in a file named AAREADME.txt in the same directory.

**Problem No. 5:** Write a program that computes the mean square error of two signals:

p05 <file1> <file2>

the mean square error = xxxxx.xxxx

The mean square error is defined as:

$$rms=\sqrt{(\frac{1}{N})\sum\_{i=0}^{N-1}(x\left[i\right]-y[i])^{2}}$$

You cannot use a library function for this – you must code it yourself. Your output must look exactly like the output above – every character must match. Why does sorting the signal (the file generated in p04) change the value of the mean square error when comparing to the unsorted file?

Put your answer in a file named AAREADME.txt in the same directory.