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 can use C/C++ or Python. But 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.

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**: Write a program that generates a binary file (-bit signed integers) containing a sinewave. The interface should be this:

p01 <amplitude> <frequency> <phase> <sample\_frequency\_in\_Hz> <duration\_in\_secs> <filename>

For example, this command line:

p01 10000 100 45 1000 10 p01.raw

produces a sinewave that follows this equation:

and writes the sample values to a binary file containing 16-bit integers. Use the “od -s” command to verify the contents of your file are correct.

**Problem No. 2:** Write a program that generates a noise signal by writing it to a binary file. The interface should be:

p02 gain <sample\_frequency\_in\_Hz> <duration\_in\_secs> <filename>

The argument “gain” controls the amplitude of the noise, and is limited to the range . A value of should produce a file of zero values. A value of should produce a signal that falls in the range . Use a uniform random number generator for this task that outputs values in the range , and scale its output to the range .

**Problem No. 3:** Write a program that computes the linear combination of two signals by adding them together:

p03 <input\_filename\_1> <input\_filename\_2> gain <output\_filename>

This command creates an output file that contains sample values that are a weighted sum of the input signals:

sig\_out[i] = sig1\_in[i] + gain \* sig2\_in[i]

Check the result with “od -s”.

**Problem No. 4:** Write a program, p04, that computes the mean and standard deviation of a signal:

p04 <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 is defined as:

The standard deviation for a sinewave is the root mean square (RMS) value: , where is the amplitude.

**Problem No. 5:** Write a program, p05, that computes the standard deviation for each “window” of a signal and displays a histogram of the values. The interface should be:

p05 <filename> <number\_of\_samples>

Your histogram should cover the range from [0, 9,999] in steps of 100.

This program converts a window of data to a standard deviation value using the following equation:

The histogram can be printed to stdout:

0,99: <value>

100,199: <value>

...

9900, 9999: <value>

I’ll test this code using your previous tools to generate noisy signals.