Functions

Spec FAQ Project Submit

Expand All Collapse All

Activation function

▼ **Which `Math` library functions may I use?**

Use `Math.exp()` and `Math.abs()`. You may also use the constants `Math.POSITIVE_INFINITY`, `Math.NEGATIVE_INFINITY`, and `Math.NaN`.

▼ **How do I detect whether *x* is NaN?**

Use `Double.isNaN(x)`. Do not write code like `(x == Double.NaN)`, as that expression evaluates to `false` for all values of `x`, including NaN.

▼ **My `softsign()` function return NaN when *x* is positive infinity (or negative infinity) instead of 1 (or –1). Why is this?**

You are dividing infinity by infinity, which leads to NaN (not a number). Include special cases for when *x* equals either `Double.POSITIVE_INFINITY` or `Double.NEGATIVE_INFINITY`,

▼ **My `tanh()` functions return NaN when *x* is larger than 750 (or smaller than –750) instead of 1 (or –1). Why is this?**

The term e^x will be larger than the largest floating-point number. As a result, you end up dividing infinity by infinity, which leads to NaN. You may use the fact that if $x \geq 20$, then `tanh(x)` should return `1.0`; if $x \leq -20$, then `tanh(x)` should return `-1.0`.

Divisors

▼ **May I use recursion to implement `gcd()`?**

That's a fine idea, but please use a loop this week. We'll learn about *recursion* next week.

▼ **How do I compute `gcd()` and `lcm()` when one (or both) of the arguments are negative?**

It follows from the definitions that $gcd(a, b) = gcd(|a|, |b|)$ and $lcm(a, b) = lcm(|a|, |b|)$.

▼ **Why does Euclid's algorithm correctly compute the greatest common divisor of two integers?**

It is based on the observation that any common divisor d of the integers a and b must also divide the differences $a - b, a - 2b, a - 3b, \ldots$, which includes $a \ \% \ b$. Here's a full explanation.

▼ **How many iterations does Euclid's algorithm take to compute the greatest common divisor of *a* and *b*?**

As you would expect, the number of iterations depends on a and b. It turns out that the smallest pair of integers that causes Euclid's algorithm to take n iterations are the consecutive Fibonacci numbers F_{n+2} and F_{n+1}. This implies that the number of steps is at most $5 \log_{10} b$, i.e., at most 5 times the number of decimal digits in the smaller number.

▼ **Why do we use the conventions that *gcd*(0, 0) = 0 and *lcm*(0, *k*) = 0?**

These are consistent with the more technical definitions of *gcd* and *lcm* as the *meet* (greatest lower bound) and *join* (least upper bound) in the lattice of divisibility.

▼ **Are there faster algorithms for computing Euler's totient function?**

Yes. Computing the totient function of an integer n is computationally equivalent to factoring n. Specifically, if the distinct prime factors of n are $p_1, p_2, \ldots, p_r$, then $\phi(n) = n\,(1 - \frac{1}{p_1})\,(1 - \frac{1}{p_2})\,(1 - \frac{1}{p_r})$. So, while computing the totient function is believed to be a computationally intractable problem for large n, it can be done substantially faster than brute-force approach adopted in this assignment.

Audio collage

▼ **What should each method do if one (or more) of the input samples is not between –1 and +1?**

That's ok. Just handle it as usual. While you should not play samples whose absolute value is greater than 1, it's fine to manipulate such values along the way.

▼ **Can `amplify()` or `mix()` produce samples whose absolute value is larger than 1, even if all of the input samples are between –1 and +1?**

Yes. While you should not play samples whose absolute value is greater than 1, it's fine to produce them as intermediate results.

▼ **How can I convert an audio file into an appropriate format for use with `StdAudio`?**

Convert it to a WAV file. Be sure to use 16-bit audio, monaural, and a sampling rate of 44,100 Hz. You may use an online conversion utility, such as Online-Convert.

▼ **What is a WAV file?**

It is popular file format for storing raw and uncompressed audio data.

▼ **The `changeSpeed()` function changes not only the speed of the sound, but also the pitch? Is there a way to change the speed without affecting the pitch?**

Speeding up the sound using resampling raises the pitch and leads to the Chipmunk effect. More sophisticated time stretching techniques are preferred in practice (such as when watching Coursera videos at 1.5× or 2× speed) because they change the speed but not the pitch.

▼ **Which other audio effects might I want to implement?**

- *clamp*: round all samples greater than +1 to +1; round all samples less than –1 to –1.

- *normalize*: rescale a sound so that all values are between –1 and +1.
- *cut*: extract a contiguous subarray from a given sound.
- *trim*: remove leading / trailing sequence of samples that are 0 (or nearly 0).
- *loop*: repeat a given sound a specified number of times.
- *mirror*: concatenate a sound with its reverse.
- *hip–hop*: increase speed of a sound; mirror it; then loop it.
- *echo, delay, reverb*: add a time-delayed version of a sound to itself, attenuated by a given factor.
- *fade-in, fade-out*: gradually increase/decrease the volume at the beginning/end of a sound.
- *crossfade*: fade-out first sound; fade-in second sound; overlap.
- *tremolo*: create a trembling effect by modulating the amplitude up and down.

You could also synthesize your own sounds by creating a *sine wave*, *square wave*, *triangle wave*, or *sawtooth wave* of a given amplitude, frequency, and duration.