**CS1101: Lab 2 – Using Structs to Build Filters and Amplifiers**

An important concept in Electrical and Computer Engineering is the design of small signal amplifiers and filters. Without electrical amplifiers, common signals such as a human pulse or the output of an electric guitar would be invisible or inaudible, respectively. Electrical filters are used to reduce noise in an environment where extremely sensitive signals are present. Therefore, amplifiers and filters are usually found together in a system. In this lab, we will be creating amplifiers and filters using structures in Scheme.

**Lab Motivation and Goals**

* Define structs
* Use structs inside other structs
* Write programs which consume structs as arguments

**Exercises**

An electrical engineer wishes to create an electrical toolbox of various types of filters. The three types of filters that he wishes to have are Butterworth, Chebyshev, and Elliptic filters. All filters have an order (number), a pass type (‘high, ‘low, ‘notch, or ‘bandpass), and a target frequency (number). Elliptic filters and Chebyshev filters also have a phenomenon called ripple; therefore, they require a ripple tolerance (number). In Chebyshev filters, ripple can occur in the pass or stop band of the filter; therefore, the band type should be recorded as well (‘pass or ‘stop).

1. Develop a data definition for the general category of filters. This definition should support all types of filters. Create a Butterworth, Chebyshev, and Elliptic filter. The Butterworth filter should have an order of 15, a pass type of ‘high, and a target frequency of 4500 (Hz). The Chebyshev filter should have an order of 10, a pass type of ‘low, a target frequency of 2000 (Hz), a ripple tolerance of 10, and ripple in the stop band. The Elliptic filter should have an order of 7, a pass type of ‘bandpass, a target frequency of 7000, and a ripple tolerance of 12.
2. Write the template for processing filters.
3. Write a function **has-pass-ripple?** that consumes a filter and produces a boolean. The output should produce true for Elliptic filters, false for Butterworth filters, and either true or false depending on the ripple type of Chebyshev filters.
4. Write a function **good-for-real-time** that consumes a filter and an order constraint (number representing the order upper limit for real time processing) and produces a symbol indicating whether the filter is good (‘good), okay (‘okay), or bad (‘bad) for real time processing. The order of the filter must be less than the input order constraint for a filter to be good for real time processing. If the order of the filter is within 5 of the input order constraint (less than or equal to the input order constraint + 5) then it is okay; otherwise, it is bad.
5. Write a function **improve-filter** that consumes a filter, a filter order adjustment, and an integer representing the percent that the ripple tolerance should be reduced and produces a filter. The output filter should have the same information as the original filter with the following adjustments:

* If the filter is Butterworth, the order should be increased by the input amount
* If the filter is Chebyshev, the order should be increased by the input amount, the ripple tolerance should be reduced by the given percent, and the band-type should be set to ‘stop.
* If the filter is Elliptic, the order should be increased by the input amount and the ripple tolerance should be reduced by the given percent.

**Everyone should be able to finish up to this point**

After the engineer has been working with filters for a while, he decides to use his filters to create amplifier systems for his toolbox. An amplifier system is composed of a filter, the peak voltage of the source wave (number), the resistance of the input wave source (number), the load resistance of the system (number), and an amplifier. Amplifiers have a name (string), an input resistance (number), a voltage gain (number), and an output resistance (number).

1. Develop a data definition for amplifiers and amplifier systems. (Make sure you provide at least one example of an amplifier system).
2. Write a function **is-good?** that takes in an amplifier and produces a boolean that indicates whether the amplifier is good or bad. The function should return true if the input resistance of the amplifier is greater than 100, the voltage gain is greater than 100, and the output resistance is less than 10. Otherwise it should return false.
3. Write a function **improve-sys** that consumes an amplifier system and produces an amplifier system. This function will increase the gain of the amplifier in the system by 100, increase the input resistance by 1000, and set the output resistance to 0 if the amplifier is bad.
4. Write a function **system-out** that consumes an amplifier system and returns the peak voltage value of the output signal from the system. This function should use a helper function to find the input voltage to the amplifier from the overall system. (Hint: In this problem Vs refers to source voltage, RIN is amplifier input resistance, Rs is source resistance, Av is amplifier voltage gain, Rload is load resistance, and ROUT is amplifier output resistance).

Equation for the actual input voltage to a system is

Equation for the output voltage from the system is

**Advanced training: Extra exercises for those who have extra time for practice.**

1. Write a function **good-filter?** that consumes an amplifier system and produces a boolean. The boolean result should be true if the order of the filter in the system is less than 20 for Butterworth, ripple tolerance is less than 10 for Elliptic, or ripple is in the stop band for Chebyshev. Otherwise, the function should produce false.
2. Write a function **pass?** that takes in an amplifier system and the frequency of the input signal to the system and produces a boolean, which is true if the signal is passed and false otherwise. A signal is passed through the filter of the system depending on the following conditions:

* Butterworth: ‘high : if frequency > target frequency + 10 then pass

‘low: if frequency < target frequency – 10 then pass

‘notch: if frequency != target frequency +/- 10 then pass

‘bandpass: if frequency = target frequency +/-10 then pass

* Chebyshev: ‘high : if frequency > target frequency + 20 then pass

‘low: if frequency < target frequency – 20 then pass

‘notch: if frequency != target frequency +/- 20 then pass

‘bandpass: if frequency = target frequency +/-20 then pass

* Elliptic: ‘high: if frequency > target frequency + 50 then pass

‘low: if frequency < target frequency – 50 then pass

‘notch: if frequency != target frequency +/- 50 then pass

‘bandpass: if frequency = target frequency +/-50 then pass

1. Write a function **replace-filter** that consumes a filter and produces a filter. The output filter should have the same parameters as the original filter, but it should be of type Butterworth. If the original filter was Butterworth, no changes should be made.
2. Write a function **replace** that consumes an amplifier system, an input resistance, a voltage gain, an output resistance, and a filter type (symbol) and produces an amplifier system. The function will replace the amplifier and filter of the system. The new amplifier will have the same name as the old amplifier and the new parameters input to the function. The new filter will be of type inputted and will have all the parameters of the old filter. Any parameters not available from the old filter should be set to 0 or ‘stop (if it’s a band type).

(Hint: Try creating a set of helper functions similar to replace-filter that will replace a filter with an Elliptic or Chebyshev filter rather than Butterworth. It may also be beneficial to create a helper function that will consume an amplifier and replace the parameters in that amplifier.)