CS116 – Lab 5
Due Date: October 10

Purpose: You have learned the basics of programming: functions, lists, loops, and conditionals. All we are going to do in this lab is shift the context from pictures to sounds. By shifting the context you will have more opportunities to master these essential concepts.

Knowledge: This lab will help you master the following content knowledge:

- How to transfer the concepts of lists, loops, and conditionals to a new context

Task: Before starting this lab, you should have read Chapters 7, 8, and 9 in your text. Follow the steps in this lab carefully and complete the assignments.

Assignment 1:

Write a function `taperEnd(sound,num)` that returns a new sound that is the same as the given sound except that the last *num* samples taper off to zero volume at a constant rate

Hint: Consider the ratio $(num - i)/num$. When i is 0 this ratio will be 1 and when i is num it will be 0. If you loop with i ranging from 0 to num, you will get the desired taper value factor from this ratio. Be careful though because the index into the sound samples is not i so you will need to figure out what index you will need.

Criteria For Success: Take a long sound like any of the bassoon files and use the explorer to look at the sound wave. There are over 50,000 sound samples in that file and most of the waves should all be the same height. If you explore the result of tapering the last 20,000 samples you should be able to clearly see the sound wave getting smaller, and smaller until it tapers down to zero at the end. Play the sound and verify that the volume tapers off at the end.

Assignment 2:

We can combine two sounds together by simply adding their values. Write a function `overlay(sound,background,start)` that overlays the sound on top of a background sound at the given starting sample location. Your method should return this new resulting sound.

There are two cases you will need to consider. Your sound overlay may be short enough so that it is entirely contained in the background length. Or your sound overlay may be long enough so that it dangles off the end of the background. In the second case, your resulting sound will need to be longer than the background sound to handle the extra sound samples after the background is finished. Make sure that your code handles both of these cases.

Whenever you are copying a "chunk" of samples, your technique should be:

1. loop with an index from 0 to the length of the chunk
2. Use `index + offset` to access the samples in the chunk where `offset` is the start location of the chunk in the sound
3. Use `index + copyOffset` to access the samples where the chunk is to be copied where `copyOffset` is the start location of where you are copying.

Criteria For Success: Test your function on both of the two cases. First try it where the overlay sound is short and the background sound is long, so that the overlay is entirely contained in the background. Explore and play the resulting sound and make sure that the overlay section contains both of the sounds played together.
Also test your function on the case where the overlay sound is longer than the background sound. Explore and play the resulting sound and make sure that it is long enough and that both sounds are played in the overlay section but only one sound or the other is played in the sections where they are not overlayed.

Assignment 3:

Write a function `stutter(sound, start, end, numRepeats)` which returns a new sound that is the same as the given sound except that the portion of the sound between *start* and *end* (inclusive) is now repeated *numRepeats* times to achieve a stuttering effect.

Criteria For Success: Test your function on a sound like `always.wav`. Select any reasonable start and end positions and a repeat values around 5 to 10. Explore and play the resulting sound and you should observe the stuttering effect.

Assignment 4:
Either in the command window or in a little function, create an audio collage. Your collage should contain multiple sounds that are transformed in several different ways using the functions you wrote here or others that are in the text. Write out your resulting sound to a file:

`s.write(filepath)`

where `filepath` is a string containing a filepath where you want the file written. A filepath will look something like "/Users/jzimmerm/Desktop/myCollage.wav"

Criteria For Success: You should have a wav file that contains your audio collage.

Submit your file containing your functions as well as your collage file. Please indicate both partner names in your submission file.