CS 1120: Media Computation Spring 2018
Lab 10: Blending sounds

To work on this lab you may want to use the sounds from the media sources zip file: (link available on course website under Resources).

Open JES and save the Program Area as lab10.py. Copy Program 110 into your Program Area and test them out as described in the textbook.

[Q1] What do you observe when you run this program on a sound?

[Q2] Which Program from chapter 6 is related to this program?

[Q3] What are the main similarities between this program and your answer to Q2?

[Q4] What are the main differences between this program and your answer to Q2?

This program is a helpful program. It allows us to layer two sounds together. Unfortunately it is specialized to only work on those two sounds. Let’s fix this.

Add to your program a function called blendTwoSounds():

```
def blendTwoSounds(sound1,sound2)
```

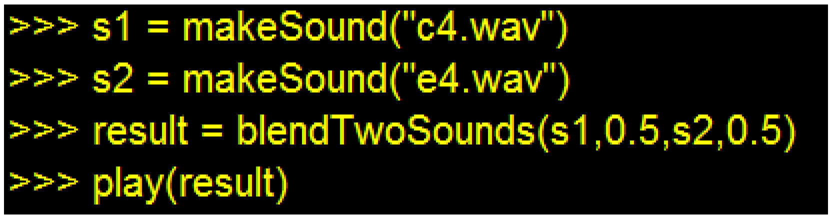
Write the code for this function so that it takes the two sounds and uses 50% of each to create a brand new, blended sound which is the exact same length as the original two sounds. To do this you should use makeEmptySound() and set it to be the length of sound1. (For ease of use at this point you may assume that sound1 and sound 2 are the exact same length and you should test it with two appropriate sounds).
It turns out that this isn't very many changes to Program 110.

[Q5] What were the key changes you made to Program 110 to use its code in this first version of blendTwoSounds()?

That result is pretty useful, but what would make it even more useful is if we could change it to not always be a 50/50 blend. Change your function to accept the following arguments:

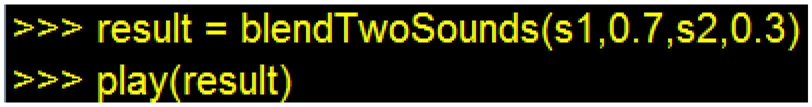
```
def blendTwoSounds(sound1, percent1, sound2, percent2)
```

Now modify your function so that it uses whatever percentage comes in when the program is used. For example, if I wrote:



```
>>> s1 = makeSound("c4.wav")
>>> s2 = makeSound("e4.wav")
>>> result = blendTwoSounds(s1,0.5,s2,0.5)
>>> play(result)
```

I would get the same result as earlier – a 50/50 blend. But if I wanted to I could write:



```
>>> result = blendTwoSounds(s1,0.7,s2,0.3)
>>> play(result)
```

This would give you 70% s1 and only 30% s2.

Test this until you feel you have it working properly.

[Q6] What were the key changes you made between the first and second versions of blendTwoSounds()?

Finally, we assumed that the two sound files would be the same length. This made it easier to decide how long to make the new empty sound file. But that might not be a legitimate assumption. Suppose that you have two sounds that you want to blend, but one is slightly longer than the other. What changes do we need to make? To help with the following questions and activity, let's consider the sounds thisisatest.wav and c4.wav. Load both of these into JES

[Q7] How many sound samples long is thisisatest.wav? How long is this in seconds (hint, length/sampling rate)?

[Q8] How many sound samples long is c4.wav? How long is that in seconds?

[Q9] If you want to blend these two sounds, how long is the output sound?

[Q10] Assume that the two sounds should start playing at the exact same time. How long will we hear thisisatest.wav and c4.wav actually overlapped?

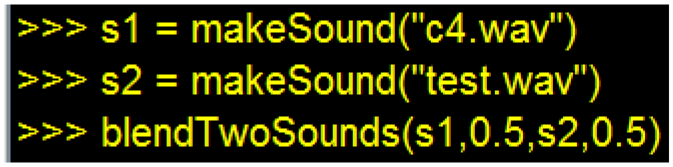
[Q11] How much time is left in the output sound at this point? (HINT: If you answered the questions correctly, the answer to this question is Q9 – Q10)

[Q12] What should we hear during this remaining time?

[Q13] If you want to blend ANY two sounds, how long is the output sound?

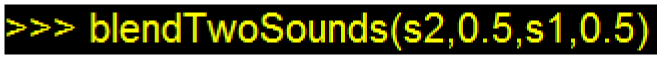
Using what you just considered, modify blendTwoSounds() one more time. This time it does not require that the two sounds be of the same length. Instead let's set the first sound to be the shorter sound and the second sound to be the longer sound. This program sets the output sound to be the length of the longer sound. It then blends the first two sounds up until the length of the shorter sound, using the percentages put it by the user. Finally, it fills the remaining sound samples with the corresponding sound samples from the longer sound at 100%.

If you get this working as described then this will work:



```
>>> s1 = makeSound("c4.wav")
>>> s2 = makeSound("test.wav")
>>> blendTwoSounds(s1,0.5,s2,0.5)
```

But this will crash:



```
>>> blendTwoSounds(s2,0.5,s1,0.5)
```

Submit your work

And you're done! You must submit two things to eLearning:

1) Your file lab10.py
2) Your answers to the questions in this lab using the response word document (you may use as many lines per question as you like).