Handing in

- Hand in a single ZIP, including:
    - all code files (.cpp and .h)
        - *in PA1 probably just a single .cpp*
    - files and scripts needed to run
        - *in PA1 **nothing** here*
    - README.txt in which is
        - What version of the asignment was solved and should be graded
        - What needs to be written in the terminal to compile and run the program
        - Anything else a stundent wishes to say (Canvas comments will **not** be delivered to the graders)
- I don't wish to ***force all-nighters upon my students*** but I feel it is their choice, so I will **not reduce** grades if submissions are LATE, as long as they are there when I wake up the next morning and collect the assignment solutions from Canvas.

In this assignment, do **not** *use pointers or dynamically allocated memory. For those students who know how to use pointers in a powerful way this limitation might make the assignment more complex but in this assignment we want to understand the structure of the values used hands on and we also wish to equalize the methods used between students of different experience. There will be plenty of opportunities to use pointers and dynamically allocated memory later in the course*.

Note that you don't have to write everything to the file at once. You can reuse a static array and write a large file in chunks. Also note that in this program your arrays may actually be enormous compared to the data in assignments you have done previously in programming courses.

This assignment can be handed in in groups of 1, 2 or 3 students.

You are given a Python script which generates a WAV file with a single sine wave, half a second long, with a frequency of 440 Hz.

PA1_WAVPort_Base.zip

Your assignment is to write this functionality in C++.

Version A - 60%

Write C++ code that compiles into a program with the exact same functionality as the Python script. Your program doesn't need to have the same functions or return values, etc. The program as a whole must simply generate the exact same file. The

file must be a WAV file with the file ending ***.wav*** and it must play correctly in VLC media player.

50% slightly diminished version: *If the file has the correct structure and plays correctly in VLC media player, but the sound is random noise or any sound other than a 0.5 second long sine wave note, the assignment will still get a passing grade of **5.0***.

Version B - 80%

Your program must ask the user what to name the output file. It is OK to assume file names with no whitespace and a maximum length of 32. The user should not enter the file ending. The program itself should add ***.wav*** to the file name.

Your program must ask the user of what frequency the note should be and of what length (in seconds) the note should be. The user should enter an integer for the frequency and a real number for the length.

The program will finally generate a WAV file with a single note, a sine wave of the given frequency and the given length.

Bonus 5%: If the program is run on the terminal with three arguments, a string, an integer and a real number then these will be used instead of asking the user for input. If not, the user must enter the input on the terminal.

Version C - 100%

The program is run on the terminal with a single argument which is the name of a text file (full name with ending and everything). This text file is of a specific format where the first line is the name of the output file (without the ending, ***.wav*** should be added), the second line is the tempo in BPM (beats per minute - integer value) and every line after that has a single letter and three integers (except silence lines that only have two integers). The letter is the name of the note, the first integer is the octave and the last two integers are the numerator and denominator for the length of the note. A lowercase letter is a base note, an uppercase letter is a sharp note. Possibilities are a/A, b/B, c/C, d/D, e/E, f/F, g/G.

The program should generate a WAV file that includes all these notes in a sequence.

Finally there should be the possibility of a silence. A silence line is an s followed by two integers, the numerator and denominator for the length. There is no octave in the silence line.

- **s 1 2**
    - A silence lasting a half note = two beats.

- **s 3 4**
    - A silence lasting a half note = three beats.
- **s 2 3**
    - A silence lasting two thirds.
    - Any integer numerator and denominator should work.
- **a 1 1 4**
    - Octave 1 A is 440 Hz, Octave 0 A is deeper, 220Hz, Octave 2 A higher, 880 Hz.
    - 1 4 means 1/4, a quarter note. ***That is exactly one beat***, so if BPM is 60, this note lasts for exactly one second. *Note that one beat is a quarter note!*
- **C 2 1 1**
    - This is an octave 2 C#
    - 1 1 means 1/1, a whole note. That's the length of four quarter notes, which is four beats.
- The note, one half note above the note *y* is *y * t*
    - Where *t = 2^(1/12)*, the twelfth root of 2.
        - Going up a half note twelve times means multiplying by 2.
            - That is a full octave.
    - The note, *x* half notes above the note *y* is *y * t^x*
    - a 1 in this case is 440 Hz.
        - You can calculate everything based on that or you can make a frequency table inside your program. Whichever works best for you.
- Here is a full octave with half note intervals:
    - **a A b c C d D e f F g G**
        - In our file format
        - **B** and **c** are the same note, also **E** and **f**
    - or
    - *A A# B C C# D D# E F F# G G#*
        - In music notation
- Your program should be able to handle 5 full octaves
    - from **a 0**
    - to **a 5**

Super bonus: **10%** on top of Version C, **5%** on top of version B

Allow for two files (or two frequencies) to be read. Mix them in the same .wav file so they harmonize.

If the first argument is -h then expect two arguments file names (version C) or two values for frequency (version B). In Version B still expect only one value for output and length.

- my_music_maker.exe -h saltkjot.txt baunir.txt
- my_note_generator -h note_file 440 659 2.3

Check out the amendments for extra information:
https://reykjavik.instructure.com/courses/4793/pages/pa1-amendments