Problem Set 9: Synthesizer II

Please send back to me via NYU Classes

- A zip archive named as
  `PS09_<your name as Last_First>.zip`
  containing the C code files that implements all aspects of all problems.

Total points: 100
Points are awarded as follows:

- **50 Points** – Port PS08 Synthesizer I code to C++
- **40 Points** – Add code to create modulator effect for synthesizer
- **10 Points** – Clear code, sensible formatting, good comments

You are given the following in PS09:

- These instructions
- build.sh which should be used to compile your C++ code.

(50 Points) Port to C++

General note: look at the code in NYU Classes: Resources/Lectures/Code/C++ for files `main.cpp, shape.cpp` and `shape.h` that show a simple example of the programming style for using `Class Shape`.

You will port the previous assignment, **PS08 Synthesizer I** to C++. This will involve the following:

- Copy all *.c and *.h code from PS08 to PS09
- Use the build.sh from PS09 to compile your code
- Rename `main.c` and `synth.c` to `main.cpp` and `synth.cpp`.
- In synth.h, change struct Synth to Class Synth.
  - Declare constructor() and destructor()
- In synth.cpp
  - The Synth constructor and destructor will be empty functions (i.e. only the ";" statement)
  - Change the definition of all member functions so that they are class members, e.g:
    `void Synth::init_synth(int in_num_chan, int in_samp_rate)`
  - In PS08, most functions defined in synth.c had as first argument a pointer to Synth. In PS09, Synth is a Class and the functions in synth.cpp are member functions of the class. Hence there is no need to have this first argument – Class member functions are aware of member variables. In a Class member function "this->" does not have to be declared and is always a pointer to the class instantiation, so when you port your code, you can replace "ps->" with "this->".
- You can compile and link *.c and associated *.h files with *.cpp files using the C++ compiler, but you need to indicate to the compiler that these are *.c files and *.h files

containing C-language function prototypes. This is done by adding the following code as the first lines of both the *.c and the *.h files.

```
#ifdef __cplusplus
extern "C"
{
#endif
```

- To match this, add the following code as the last lines of the *.c and *.h files:

```
#ifdef __cplusplus
}
#endif
```

Do this in `key_tables.c, key_tables.h, paUtils.c` and `paUtils.h.`

After these changes, the program should compile and run in a manner identical to PS08.

(40 Points) Add Modulator Effect

Add a modulator whose frequency is read from the command line.
in **main.cpp** declare

```
double mod_freq;
```

The frequency of the modulator will be an optional first command line argument. If the argument is NOT present, use a default value of zero for the modulator frequency. If the first argument is "—help" print a usage line as follows:

```
fprintf(stderr, "Usage: %s [modulator_freq]\n", argv[0]);
```

where the square brackets indicate that the argument is optional.

If the first argument is present, use `atof()` to set `mod_freq` from the first arg. Also, check that

```
mod_freq > 1 && mod_freq < 4000
```

If not, print an error message and return -1;

Modify Class Synth to have a new struct. Add this code below the declaration of the Tone struct:

```
/* modulator */
typedef struct {
    double freq;
    double phase_inc;
    double phase;
} Mod;
```

In Class Synth below

```
Tone tone[KEYS_VOICED];
```

Define an instantiation of Mod

```
Mod mod;
```

Add a new member function to Class Synth, in synth.h and synth.cpp:

```
Synth::set_mod(double in_freq);
```

It sets mod.freq = in_freq and mod.phase = 0. If in_freq is 0, then it sets

```
mod.phase_inc = -1;
```

which will disable the modulator. Otherwise it sets `phase_inc` to generate a sine tone in the same manner as in the Synth tone synthesizer.

Finally, at the end of member function `synth_block()`, add the code to perform the modulation:

```
/* modulation */
if (this->mod.phase_inc > 0) {
     v *= sin(this->mod.phase);
     this->mod.phase += this->mod.phase_inc;
}
```

Now when you execute

```
./synth 20
```

Or

```
./synth 2000
```

You will get an interesting effect!