Computer Science 327
Project 2, Part a
C++ Programming Project
Digital Sound Cont.

The Assignment (Part a)

In this part of the assignment we will create a C++ object oriented version of the sound toolkit. Please make a new repository named "coms327p2" and make sure Jim and all the TAs are added as maintainers.

1. (100 points) In this first part of the assignment you will create a class that encapsulates a sequence of sound samples. Note that we will modify this class later in the assignment, but for now it must satisfy the following requirements.

    a. The class is named "SoundSamples" and is defined in "SoundSamples.h" and implemented in "SoundSamples.cpp"

    b. The class contains properties (member variables) that define the sample rate, a sequence of samples, and the number of samples in the sequence.

    c. The class provides a default constructor that creates an object that represents a sequence of samples of length 0 at sample rate 8000.

    d. The class provides a specific constructor with the signature prototype:
    `SoundSamples( float *, int, float);`
    where the first parameter is an array of floats that represents the samples, the second parameter is the number of samples in the array, and the third parameter is the sample rate. Note that the array of samples must be deep copied. Do not just assign the pointer.

    e. The class provides a specific constructor that takes two **int** parameters. The fist parameter is the number of samples, and the second parameter is the sample rate. The constructor creates a sequence of samples of the specified length that are all zeros with the specified sample rate.

    f. The class provides a getter function for the sample rate.

    g. A correct copy constructor that deep copies the object. This again provides a deep copy.

    h. A correct overloaded assignment = operator. This also is a deep copy.

    i. A "length" method that returns the number of samples in the sample array.

j. Overloaded operator [] that returns a reference to the sample at a specific index. This should allow you to statements like a[4]=0.5; which would assign sample at index 4 to 0.5.

k. Overloaded operator + that appends one SoundSample to another. For example: x = y + z should append the samples of z at the end of the samples for y.

2. (75 points) Create a class that represents a wave. The name of the class is "Wave" and must be defined in the file "wave.h" and implemented in "wave.cpp" This class is **immutable** and encapsulates the following information.
   a. Name: This is a unique name that identifies the particular wave.

   The class also contains the method "generateSamples" that creates the samples for the specified wave at the specified sample rate for the specified duration. The prototype for the method is:

   `SoundSamples *` generateSamples(float frequency, float samplerate, float duration);

   This method calls "generateFunction" which resides in a subclass that will generate a sample of a wave given the time.

   This means that your class.h file contains the following prototype for the class.

```
virtual float generateFunction(float time)=0;
```

   This is the equivalent of an abstract function in Java. The idea is that you subclass the wave class and implement the generateFunction that will produce different waves. This will be discussed extensively in class when inheritance is covered. For example, you can implement a sine wave with the following class.

```
class SineWave : public Wave
{
      generateFunction( float time )
      {
            // code to compute sample at time
            //return computed sample
      }
}
```

   Note that this will need a constructor that takes in the name of the wave and then calls the appropriate super constructor. You can then use code like this to generate a sine wave.

```
Wave *w = new SineWave( "MySineWave" );
SoundSamples *s = w->generateSamples( 440, 8000, 2.5 );
```

3. (50 points) Create four classes SineWave, SquareWave, TriangleWave, SawtoothWave that are all a subclasses of Wave that implements the generateFunction appropriately for the wave. Make sure that your constructor can set a name of the wave.

4. (50 points) Create a class named “SoundIO” (in files soundio.h and soundio.cpp) that contains the static function
   `OutputSound(SoundSamples *samples, string filename );`
   You must use **ofstream** to write to the file.

5. (50 points) Write a main function in main.cpp that inputs from the user a wave type (1, 2, 3, or 4) for the type of file as defined above, a sample rate, frequency, duration, and file name, in that order. The program then outputs the samples for the wave to the specified file.

6. (25 points) Create an appropriate makefile.