```cpp
/*****************************************************************************/
/* main.cpp */
/* Yoo Min Cha */
/* PS5b */
/* Professor Martin */
/* 30 March 2014 */
/*****************************************************************************/

// compile with
// make

#include <iostream>
#include <string>
#include <exception>
#include <stdexcept>
#define _USE_MATH_DEFINES
#include <math.h>
#include <limits.h>
#include <vector>

#include <SFML/Graphics.hpp>
#include <SFML/System.hpp>
#include <SFML/Audio.hpp>
#include <SFML/Window.hpp>

#include "RingBuffer.hpp"
#include "GuitarString.hpp"

using namespace std;

#define CONCERT_A 440.0
#define SAMPLES_PER_SEC 44100

// this makes a vector of <sf::Int16> from the Karplus-Strong string simulation
vector<sf::Int16> makeSamplesFromString(GuitarString gs) {
  vector<sf::Int16> samples;

  gs.pluck();
  int duration = 8; // seconds
  int i;
  for (i= 0; i < SAMPLES_PER_SEC * duration; i++) {
    gs.tic();
    samples.push_back(gs.sample());
  }

  return samples;
}

int main()
{
  sf::RenderWindow window(sf::VideoMode(300, 200), "SFML Guitar Hero Lite");
  sf::Event event;

  sf::Sound sounds[37];
  sf::SoundBuffer bufs[37];

```

```
 string keyboard = "q2we4r5ty7u8i9op-[=zxdcfvgbnjmk,.;/’ ";
 int posMap[128];
 for(int i=0;i<128;i++) posMap[i]=-1;

 for(int i=0;i<keyboard.size();i++){
  posMap[keyboard[i]] = i;
 }

 for(int i=0;i<37;i++){
  double freq = CONCERT_A*pow(2, ((double)(i-24)/12.0));
  GuitarString gs = GuitarString(freq);
   /*sf::Sound sound;
    sf::SoundBuffer buf;
    sounds.push_back(sound);
    bufs.push_back(buf);
    */
  vector<sf::Int16> samples = makeSamplesFromString(gs);
  if (!bufs[i].loadFromSamples(&samples[0], samples.size(), 2, SAMPLES_PER_SEC))
   throw std::runtime_error("sf::SoundBuffer: failed to load from samples.");
    sounds[i].setBuffer(bufs[i]);
  }

 while (window.isOpen()) {

  while (window.pollEvent(event)) {

   switch (event.type) {
   case sf::Event::Closed:
    window.close();
    break;

   case sf::Event::TextEntered:
    cout << static_cast<char>(event.text.unicode) << " "
      << posMap[static_cast<char>(event.text.unicode)] << endl;

    if (posMap[static_cast<char>(event.text.unicode)]>=0)
      sounds[posMap[static_cast<char>(event.text.unicode)]].play();
    break;

   default:
    break;

   }

   window.clear();
   window.display();

  }
 }
 return 0;
}
```