The **IRremote** library and the Arduino built in **tone** library have conflicts due to both libraries are using the same hardware interrupt timer 2.

This **NewTone** library uses hardware interrupt timer 1 and can be installed in the "Arduino-1.x\libraries\" folder of your PC to work around the timer conflicts.

Function Syntax:
NewTone(pin, frequency, duration);

<u>Homework 9 Pseudo Code:</u>

Include the <IRremote.h> and <NewTone.h> libraries.

Define the following **symbolic constants** (reference Chapter 13 lecture slides #4 and #5) for your remote control buttons IR decoded values (i.e. 0xff629d, 0xff906f, and so on…) for **MODE**, **PLUS**, **MINUS**, **ZERO**, **DOOR**, **USD**, **ONE**, **TWO**, **THREE**, **FOUR**, **FIVE**, **SIX**, **SEVEN**, **EIGHT**, and **NINE**.

(Declare function prototypes for the following **programmer defined functions**, reference Chapter 5 lecture slides #7-#14)
Declare a function prototype for a function named **lightsOff**. The function takes no input and provides no output.
Declare a function prototype to **playPiano**. The function takes **one unsigned long** input parameter, **key**, that represents the decoded hex value of the button pressed from the remote control. The function provides **no output** argument.
Declare a function prototype to **simulateDoor**. The function takes **one unsigned long** input parameter, **key**, that represents the decoded hex value of the button pressed from the remote control. The function provides **no output** argument.
Declare a function prototype to **showVolume**. The function takes **one unsigned long** input parameter, **key**, that represents the decoded hex value of the button pressed from the remote control. The function provides **no output** argument.
Declare another function prototype for a function to **playMusic**. The playMusic function has **no input** and **no output** arguments.

(Declare the following global variables. Reference Storage Classes and Scope from Chapter 5 lecture slides #25 - #33.)
Declare a constant integer variable, **recPin**, and initialize the value to 2.
Declare a constant integer variable, **buzPin**, and initialize the value to 7.
Declare a constant array of 9 integers, **ledPin**, and initialize their values to 0, 3, 4, 5, 6, 8, 9, 10, 11.
Declare a constant array of 13 integers, **piano**, and initialize their values to 0, 1047, 1109, 1175, 1245, and so on…

(Reference sample code from the Homework 9 assignment to capture and decode the signals from the remote control.)
IRrecv irrecv(**recPin**);
decode_results **result**;

```
Declare an integer variable, modePress, and initialize the value to 0.
Declare an integer variable, pianoMode, and initialize the value to 0.
Declare an integer variable, garageMode, and initialize the value to 0.
Declare an integer variable, volumeMode, and initialize the value to 0.
Declare an unsigned integer variable, numberOfLights, and initialize the value to 0.
Declare an unsigned long variable, startTime, and initialize the value to 0.
Declare an unsigned long variable, currentTime, and initialize the value to 0.
Declare an integer variable, moving, and initialize the value to 0.
Declare an integer variable, moveUp, and initialize the value to 0.

(The setup function runs once when you press reset.)
void setup()
{
  Set serial baud rate for the Serial Monitor window to 9600 for debugging.
  Enable the IR receiver by calling the IR library function irrecv.enableIRIn().
  for (loop integer i from 1 to 8)
  {
    Call the Arduino built-in pinMode function and provide ledPin array index i and OUTPUT as function inputs.
  }
}

(The loop function runs over and over again forever.)
void loop()
{
  Declare a local unsigned long variable, irVal.

  If (the value of irrecv.decode(&result) is a non-zero value)
  {
    Save the result.value to irVal.
    Call the Arduino Serial.println() function and provide irVale and HEX  as function input for debugging
purpose.

    If (the value of irVal is equal to MODE)
      Set modePress to 1.
    Else if (modePress is a non-zero value)
    {
      Switch (based on the value of irVal)
      {
        Case ONE:
          Set modePress to 0.
          Set pianoMode to 1.
          Set garageMode to 0.
          Set volumeMode to 0.
          Call the lightsOff function.
          Set **numberOfLights** to 0.
        Case **TWO**:
          Set **modePress** to 0.
          Set **pianoMode** to 0.
          Set **garageMode** to 1.
          Set **volumeMode** to 0.
          Call the **lightsOff** function.
          Set **numberOfLights** to 0.
        Case **THREE**:
          Set **modePress** to 0.
          Set **pianoMode** to 0.
          Set **garageMode** to 0.
          Set **volumeMode** to 1.
          Call the **lightsOff** function.
          Set **numberOfLights** to 0.
        Case **FOUR**:
          Set **modePress** to 0.
          Set **pianoMode** to 0.
          Set **garageMode** to 0.
          Set **volumeMode** to 0.
          Call the **lightsOff** function.
          Set **numberOfLights** to 0.
          Call the **playMusic** function.
        In the default case: just break.
      }  // end of switch
    }  // end of else if the mode button has been pressed
    Else if (**pianoMode** is a non-zero value)
    {
      Call the **Serial.println** function and print "In piano mode!" for debugging purpose.
      Call the **playPiano** function and provide **irVal** as the function input.
    }  // end of else if in piano mode
    Else if (**garageMode** is a non-zero value)
    {
      Call the **Serial.println** function and print "In garage mode!" for debugging purpose.
      Call the **simulateDoor** function and provide **irVal** as the function input.
    }
    Else if (**volumeMode** is a non-zero value)
    {
      Call the **Serial.println** function and print "In volume mode!" for debugging purpose.
      Call the **showVolume** function and provide **irVal** as the function input.
    }
    Call the **irrecv.resume**() function to resume IR receive.
  }  // end of if IR sensor get any remote control signal
```

```
  Else if (garageMode is equal to 1 AND moving is equal to 1)
  {
    If (moveUp is a non-zero value)
      Call the NewTone function and provide buzPin, 466, and 50 as the function inputs.
    Else
      Call the NewTone function and provide buzPin, 156, and 50 as the function inputs.
    Call the Arduino built-in delay function and provide 50 as the function input.

    Call the Arduino built-in millis function and save the return value in currentTime.
    If ((currentTime minus startTime) is larger or equal to 1000)
    {
      Modify startTime to the value of currentTime.
      If (the value of numberOfLights is equal to 0 AND the value of moveUp is equal to 0)
      {
        Set moving to 0.
        Set moveUp t0 1.
      }
      Else if (the value of numberOfLights is equal to 8 AND the value of moveUp is a non-zero value)
      {
        Set moving to 0.
        Set moveUp to 0.
      }
      Else if (moveUp is a non-zero value)
        Increment numberOfLights by 1.
      Else
        Decrement numberOfLights by 1.

      Call the Arduino built-in Serial.print function and print "Number of lights: " in the debug window.
      Call the Arduino built-in Serial.println function and provide numberOfLights as the function input.
      Call the lightsOff function.
      For (loop integer j from 1 to numberOfLights)
        Call Arduino built-in digitalWrite function and provide ledPin array index j and HIGH as function inputs.

    }  // end of if 1 second passed
  }  // end of else if the garage mode and door is moving
  Call the Arduino delay() function and provide 100 ms as function input.
}  // end of loop

void lightsOff()
{
  For (loop integer i from 1 to 8)
    Call Arduino built-in digitalWrite function and provide ledPin array index i and LOW as function inputs.
}
```

```
void playPiano(unsigned long key)
{
  Declare a local integer variable noteNum and initialize the value to 0.
  Declare a local integer variable light1Num and initialize the value to 0.
  Declare a local integer variable light2Num and initialize the value to 0

  Switch (based on the value of key)
  {
    Case ZERO:
      Set noteNum to 1.
      Set light1Num to 1.
      Set light2Num to 1.
    Case DOOR:
      Set noteNum to 2.
      Set light1Num to 1.
      Set light2Num to 2.
    Case USD:
      Set noteNum to 3.
      Set light1Num to 2.
      Set light2Num to 2.
    Case ONE:
      Set noteNum to 4.
      Set light1Num to 2.
      Set light2Num to 3.
    Case TWO:
      Set noteNum to 5.
      Set light1Num to 3.
      Set light2Num to 3.
    Case THREE:
      Set noteNum to 6.
      Set light1Num to 4.
      Set light2Num to 4.
    Case FOUR:
      Set noteNum to 7.
      Set light1Num to 4.
      Set light2Num to 5.
    Case FIVE:
      Set noteNum to 8.
      Set light1Num to 5.
      Set light2Num to 5.
    Case SIX:
      Set noteNum to 9.
      Set light1Num to 5.
      Set **light2Num** to 6.
    Case **SEVEN**:
      Set **noteNum** to 10.
      Set **light1Num** to 6.
      Set **light2Num** to 6.
    Case **EIGHT**:
      Set **noteNum** to 11.
      Set **light1Num** to 6.
      Set **light2Num** to 7.
    Case **NINE**:
      Set **noteNum** to 12.
      Set **light1Num** to 7.
      Set **light2Num** to 7.
    In the default case, just break.
  }  // end of switch statement
  Call the Arduino built-in **digitalWrite** function and provide **ledPin** array index **light1Num** and **HIGH** as inputs.
  Call the Arduino built-in **digitalWrite** function and provide **ledPin** array index **light2Num** and **HIGH** as inputs.
  Call the **NewTone** function and provide **buzPin**, **piano** array index **noteNum**, and 500 as inputs.
  Call the Arduino built-in **delay** function and provide 500 as function input.
  Call the **lightsOff** function.
}  // end of the playPiano function

void **simulateDoor**(unsigned long **key**)
{
  If (the value of **key** is equal to **DOOR**)
  {
    If (the value of **numberOfLights** is equal to 0)
    {
      Call the Arduino built-in **millis** function and save the function return value to **startTime**.
      Set **moveUp** to 1.
      Set **moving** to 1.
    }
    Else if (the value of **numberOfLights** is equal to 8)
    {
      Set **moveUp** to 0.
      Set **moving** to 1.
    }
    Else if (**numberOfLights** is larger than 0 AND **numberOfLights** is less than 8)
    {
      If (**moving** is a non-zero value AND **moveUp** is a non-zero value)
      {
        Set **moving** to 0.
        Set **moveUp** to 0.
      }
      Else if (moving is a non-zero value AND moveUp is equal to zero)
      {
        Set moving to 0.
        Set moveUp to 1.
      }
      Else if (the value of moving is equal to 0)
      {
        Set moving to 1.
      }
    }
    Else
      Call the Serial.println function to print "Error counting lights!" in the debug window.
  }  // end of if garage door key pressed
}  // end of the simulateDoor function

void showVolume (unsigned long key)
{
  If (the value of key is equal to PLUS AND the value of numberOfLights is less than 8)
    Increment numberOfLights by 1.
  Else if (the value of key is equal to MINUS AND the value of numberOfLights is larger than 0)
    Decrement numberOfLights by 1.

  Call the Arduino built-in Serial.print function to print "Number of lights: " in the debug window.
  Call the Arduino built-in Serial.println function and provide numberOfLights as the function input.
  Call the lightsOff function.
  For (loop integer i from 1 to numberOfLights)
    Call the Arduino built-in digitalWrite function and provide ledPin array index i and HIGH as function inputs.
}  // end of showVolume function

Void playMusic()
{
  // Feel free to re-use your song and light show here
}
```