ECE 177 Lab 11 Spring 2018

Purpose: More practice with arrays & pointers. Introduction to "`structs`" and serial I/O.

Assignment: For this lab your program will get a key from the keypad and then play one of two songs depending on if a '1' or '2' key is hit. It will then loop back to get another key. For the "A" grade you will hook up a serial device and will listen for characters (typed on the PC), echo them (i.e., send them back to the PC), and then play one of two songs depending on if a '1' or '2' key is received.

Prelab: CodeLab problems related to this lab must be completed before coming to lab.

Notes:
The web page gives partial code including several `struct` and function declarations that you will use in your program. Use as given.

There are two `structs` shown: a note and a song. The `note` gives the information for a note, specifically the note number (index into a frequency table), and duration in 1/8ths of a second. A song gives the information for a song, specifically the number of notes, and an array of the actual notes. Note that the array is declared `const`. This places it in program memory (ROM) which is more plentiful than data memory (file registers – RAM). Also note that there is a limit of 30 characters per song.

Two songs and a frequency table are also given. The frequency table gives the frequency in cycles per second for each note number. (E.g., note number 0 has a frequency of 262 Hz.)

The speaker will be hooked up to PORTC pin 1. Remember to make this pin an output, but do this after calling "`lcd_init()`", as that function might possibly reset DDRC bits. To make sound you will toggle this pin up and down. E.g., to play a tone of 440Hz, toggle this pin 880 times per second (440 complete cycles). Create a "`#define TOGGLESPEAKER XXX`" at the top of your code that will toggle this bit.

Your main program will, in an infinite loop, wait for a key to be entered on the keypad and then if the key was '1' then play the Stein Song, and if it is '2', play "Mary Had a Little Lamb". Then loop back and do it again.

In addition to functions previously written you will need:

`playnote()` will play one note: First compute the number of times to toggle the speaker (full cycles – low then high). This is given by the frequency times the note duration (divide by 8 as duration is given in 1/8ths of a second.) Then compute the time between toggles. This is half of 1,000,000 divided by the frequency. Do math as best possible to avoid round-off errors. Then toggle the speaker the computed number of times. For each toggle, drive the speaker low, delay the computed time, drive the speaker high, delay the computed time. Delay 10ms at the end to provide a slight break between notes.

`playsong()` will simply play the notes of a song (passed to it as an argument) using a loop.

Doing the above will get you the “B” grade. For the “A” grade, first get the “B” grade checked off as well as any previous labs, as you will need to add a serial device. A-grade code will receive characters from the keyboard, echo them back to the PC. The '1' and '2' ASCII characters will select a song as before, but also the ‘z’ character will switch you to “piano” mode. In this mode each digit pressed will play a corresponding note for 1/8 second. (`q’, `w’, `e’, `r’, `t’, ‘y’, ‘u’, ‘i’, ‘o’ correspond to notes C, D, E, F, G, A, B, C, D, respectively and ‘2’, `3’, ‘5’, ‘6’, ‘7’, ‘9’ correspond to the notes C#, D#, F#, G#, A#, C#, respectively). `playnote()` will suppress its 10ms delay while in this mode. Entering ‘z’ again will terminate this mode. You will need the following functions:

A grade details:

`serialInit(void)` will initialize the serial port. It has been written for you, so use it verbatim.

`serialRead(void)` will receive a character from the serial port. It works as follows: wait for the `RXC0` bit in the `UCSR0A` to go to 1, then read the character in `UDR0` and return it.

`serialWrite(unsigned char c)` will transmit a character out the serial port. It works as follows: wait for the `UDRE0` bit in the `UCSR0A` to go to 1, then write the character to `UDR0`.

NOTE: `RXC0` and `UDRE0` are each defined in the header file to be the bit number of the corresponding bit.

NOTE: initialization in your main routine should make the lower two bits of PORTC and PORTD to be inputs, taking care not to change the other bits (some are used by the LCD). Then call `serialInit()`.

NOTE: the speaker will have to be moved to one of PORTD pins 2 through 7.

NOTE: For wiring of the serial device: Place the device at the far left of the breadboard, with pins 1 and 2 to the right overlapping two LED pins, and pins 15 to 28 hanging off the edge of the board. (See TA for help.) Device pin1 to AVR pin 14, Device pin 5 to AVR pin 15, and Device pin 7 to GND.