ECE 177 Lab 9 Spring 2019

Purpose: Using arrays. More programming practice.

Assignment: For this lab you will again connect a "hex" keypad to your AVR and you will write a "password entry" program. The program will wait for a user to enter a password on the keypad and will then display either “Open Lock” for two seconds if correct or it will blink “Error” 4 times if incorrect. It will then loop back and wait for another password entry. Your password will be stored in an array inside your program.

Notes:
Your program should use five functions: `getkey()`, `mapkey()`, `getpw()`, `compareArrays()` and `mydelayms()`. See the prototypes for these functions in the starter code.

Use your **`getkey()`** and **`mapkey()`** verbatim from the previous lab – no changes.

Function **`getpw()`** will record keys pressed on the keypad until a ‘#’ key is hit (much like the previous lab). As before ‘*’ will be used as a “backspace” key. Other keys are stored as returned by “`mapkey()`” (digits are stored as 0-9 and ‘A’ through ‘D’ are stored as 10 through 13). The keys are stored in an array (buffer) that is passed to the routine as the second argument. The first argument to the function is the maximum size of the array. Any digit or letter keys hit when the buffer is full will be ignored. The function will return the number of values in the buffer. The function can operate as follows: In a loop, wait for all keys to be released, wait for a key to be pressed, call `mapkey()` with that key. If the mapped key is a digit or letter, then add it to the buffer if the buffer is not full (ignore it otherwise). If the mapped key is an ‘*’ then reduce the current size of the buffer by one (but don’t make it negative size). If the mapped key is a “#” then return the size of the buffer.

Function **`compareArrays()`** will compare two arrays and will return 1 if the arrays are equal, and 0 otherwise. In addition to the two arrays passed as arguments, a third argument will be the size of the arrays. The function should also work correctly if the size is given as zero.

Function **`mydelayms()`** will simply delay the given number of milliseconds. For this routine, Use a pair of nested `for` loops, the inner one delays one millisecond and the outer one causes this one millisecond delay to happen the desired number of times. Adjust your inner delay so it delays 1 ms. All delays in your main program will use **`mydelayms()`**.

Your main function will include (as local variables) an array which stores the correct password and a variable that indicates the password length. Your code should be written in such a way that changing this length and the array is all that is needed to make a corresponding change in the password. Your main function will also include an array (buffer) to hold the password entered by the user. Your main code will (in an infinite loop): call `getpw()` to get a password entered by the user. Then, if the password is the correct length (check the return value of `getpw()`) and if the password is correct (call `compareArrays()`) then display “Open lock” for two seconds. If it is incorrect, then blink “Error” four times in the display over a two second period.

Doing the above will get you a "B" grade. For an "A" have your code print an ‘*’ for every character entered in the buffer. Remove one ‘*’ when the “Backspace” key (‘*’) is hit.
Prelab: CodeLab problems related to this lab must be completed before coming to lab.