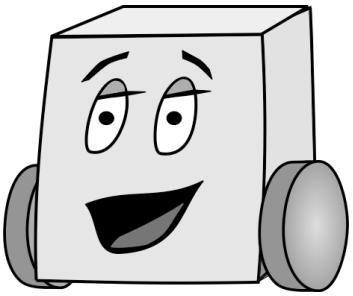

E11: Autonomous Vehicles

Fall 2014

PS 2: Music and Memory Game

In this problem set you will write two programs: one will play music on your Mudduino and the other will play a memory game. These programs will help you become comfortable using arrays, control statements, and function calls in C.

Important: keep track of how long it takes for each program by placing the following comment on the first line of each program file (where xx is the number of hours that it took you):

```
// Time to complete program = xx hours
```

Part 1: Playing a Song on the Mudduino

Program your Mudduino to play the sequence of notes – specified as a frequency and duration – shown in Table 1. Your program should include a 30 ms delay between each note (frequency) that is played. Name this sketch ps2_1_Lastname_Firstname.

Frequency	Duration (ms)
294	220
294	220
392	970
587	470
523	135
494	135
440	135
784	970
587	470
523	135
494	135
440	135
784	970
587	470
523	135
494	135
523	135
440	1470

Table 1. Notes to play on your Mudduino

You may find these two arrays useful:

```
int frequency[18] = {294, 294, 392, 587, 523, 494, 440,
784, 587, 523, 494, 440, 784, 587, 523, 494, 523, 440};

int duration[18] = {220, 220, 970, 470, 135, 135, 135, 970,
470, 135, 135, 135, 970, 470, 135, 135, 135, 1470};
```

Can you name the song? After you have successfully programmed and played the song, include the name of the song at the top of the program you submit.

Part 2: Memory Game

In the final part of this assignment, you will implement a Memory Game similar in spirit to Simon (see http://en.wikipedia.org/wiki/Simon_(game)) – call this Arduino sketch ps2_2_Lastname_Firstname.

The game should be played as follows:

1. First, your Mudduino should create 8 random LED flashes, some of them green and some of them red.

2. Then, the user should enter 8 keypresses into the Mudduino – this should be an attempt to replicate the LED flashes. For convention, let's let 1 represent green LED flashes and let 0 represent red LED flashes.

3. Then, the Mudduino should compute the user's *score* and print it out to the user. The program should compute the score by comparing the record of the correct LED flashes (1s and 0s) with the keypresses the user entered (1s and 0s). The score will be the *correlation* between the two sequences: for each keypress that is correct, the user gets 1 point; for each incorrect keypress, the user gets -1 point. The *score* is the sum of the 8 keypress scores. For example, for the following 8-bit binary sequences, the total score is 4:

```
LED sequence:   0   1   1   0   0   1   1   0
User keypresses: 0   1   0   0   0   1   1   1
                ----------------------
                +1  +1  -1  +1  +1  +1  +1  -1 = 4
```

4. Finally, the Mudduino should ask the user to hit a key before playing again.

Write the following four functions for each of the four steps, respectively:

```
void generateLEDFlashes();
void getUserResponse();
void correlate();
void promptUser();
```

Note: Make sure that in your final version, the sequence is truly random using the `randomSeed()` command. Have fun!

Deliverables

You are responsible for turning in 2 Arduino files to the "Resources/Problem Set 2" folder in the E11 page on Sakai:

- `PS2_1_Lastname_Firstname.ino`
- `PS2_2_Lastname_Firstname.ino`

The files are due before class.

Grading

Your code will be graded as follows

- 0.5 points for each program that compiles
- 0.5 additional points for each program that works according to the requirements described above.
- 1.0 additional points for your 3 programs being commented
- This results in 3.0 points maximum