ME133 Lab5 Hints

Tones are achieved using the Arduino tone function. This function takes parameters in the form tone(int pin, int frequency, int duration). A mapping of frequencies to music pitch notation was taken from the Arduino tutorial library [1]. Short melodies (tunes) are implemented as arrays with the following format:

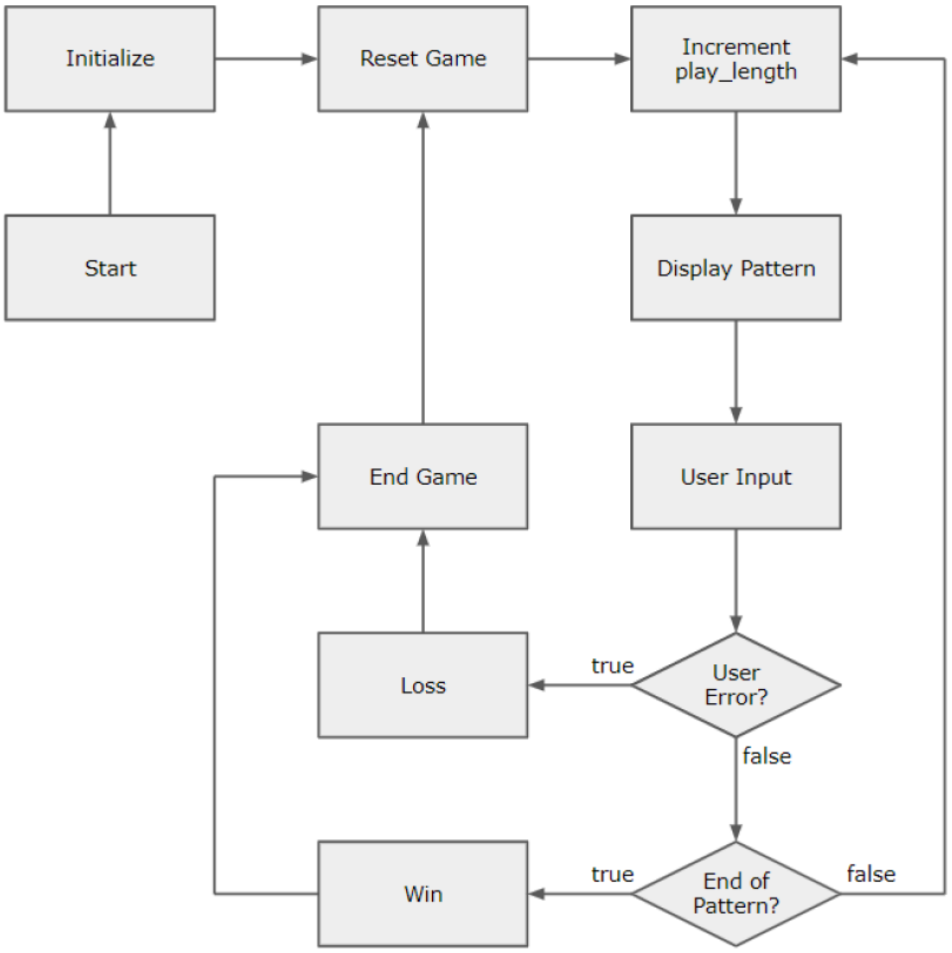
```
// tunes are organized like this:
// [
// note_count, bpm,
// note_0_pitch, note_0_duration,
// note_1_pitch, note_1_duration,
// note_2_pitch, note_2_duration,
// ...
// note_last_pitch, note_last_duration
// ]
const int tune_bb_organ[ ] = {
7, 240,
NOTE_C4, 2,
NOTE_C5, 1,
NOTE_A4, 1,
NOTE_G4, 1,
NOTE_E4, 1,
NOTE_G4, 3,
NOTE_D4, 3
};
```

Where bpm is beats per minute. The values of NOTE_X were taken from the tutorial mentioned above. The duration of the notes is in terms of the number of quarter notes at a particular bpm. The function void play_tune(int pin, int tune[ ]) interprets these arrays and writes the tune to the specified pin.

```
// plays a tune based on the comment above to the specified pin
// note: when calling this function, use the cast (int *) on the tune
// the cast (int *) is used to remove a warning that the tune is being initialized as a non-const
copy
// https://forum.arduino.cc/t/passing-constant-text-to-a-function/524005
void play_tune(int pin, int tune[]) {
  int tuneLength = tune[0];
  int bpm = tune[1];
  float unit_time = 60./bpm * 1000.;
  for (int i = 2; i <   (tuneLength+1)*2; i += 2) {
    float notePitch = tune[i];
```

```
        float noteDuration = unit_time * tune[i+1];
        tone(pin, notePitch, noteDuration);
        // to distinguish the notes, set a minimum time between them.
        // the note's duration + 30% seems to work well:
        int pauseBetweenNotes = noteDuration * 1.30;
        delay(pauseBetweenNotes);
        noTone(pin);
    }
}
```

The game is implemented in a series of states. Figure below shows a state flow diagram of the game.



Arduino functions you may use:
random()
unsigned long()
bool()