4	Start a timer, and wait for the player to press a button. The player has 1.5 seconds to press the correct button. **A:** If the time limit runs out before a button is pressed, the player loses. **B:** If the player presses the wrong button, the player loses. **C:** If the player presses the right button, move on to the next number in the sequence. **D:** Repeat this process until the player has lost or correctly repeated the sequence for this round.
5	If the player repeats the entire sequence for that round, increase the round number by one (this will add one extra item to the end of the pattern). Then go back to step 3.
6	Keep incrementing the round until the player loses or finishes 10 rounds. If the player finishes 10 rounds, play the winning sequence.

CODE TO NOTE

ELAPSED TIME:

```
millis();
```

The `millis()` function returns the number of milliseconds that have passed since the RedBoard was last turned on.

BOOLEAN VARIABLES:

```
boolean variable_name;
```

The name for these variables comes from Boolean logic. The `boolean` variable type only has two values: 1 or 0 (also known as HIGH or LOW, ON or OFF, TRUE or FALSE). Using Boolean variables helps save memory on your microcontroller if you only need to know if something is true or false. Space in your microcontroller's memory is reserved when a variable is declared. How much memory is reserved depends on the type of variable.

STORING PIN NUMBERS IN ARRAYS:

```
int led[ ] = {3,5,7,9};
```

Sometimes you will want to cycle through all of the LEDs or buttons connected to a project. You can do this by storing a sequence of pin numbers in an array. The advantage of having pins in an array instead of a sequence of variables is that you can use a loop to easily cycle through each pin.

FUNCTIONS TO NOTE

```
flashLED(#LED to flash);
```

This turns one of the four LEDs on and plays the tone associated with it.
0 = Red, 1 = Yellow, 2 = Green, 3 = Blue.