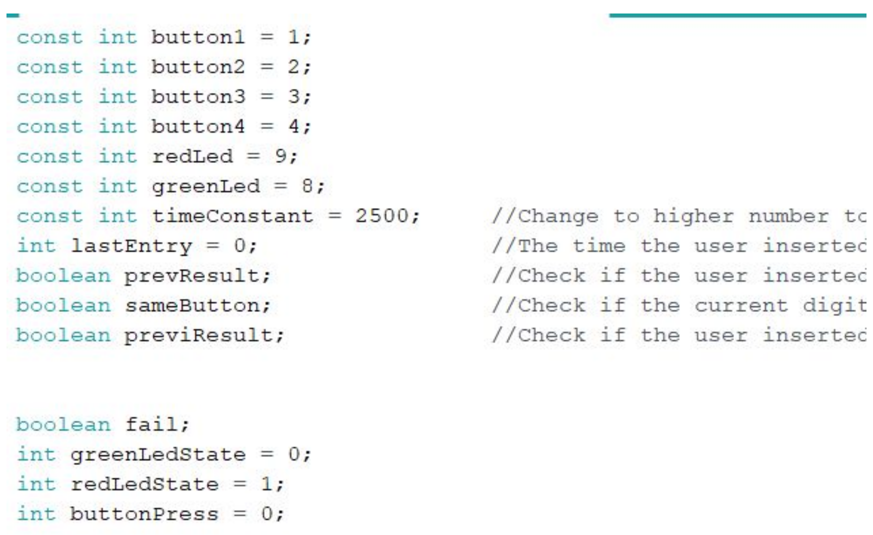
Button Combination Lock

<u>Introduction:</u>

- const int: Is used to identify where the button (red, blue, yellow, green) are on the Arduino pin board. Using const and int allows a name to be associated with a variable.
    - const: variable qualifier that modifies the behavior of the variable, making a variable "read-only".
    - Int (Integers) are your primary data-type for number storage.



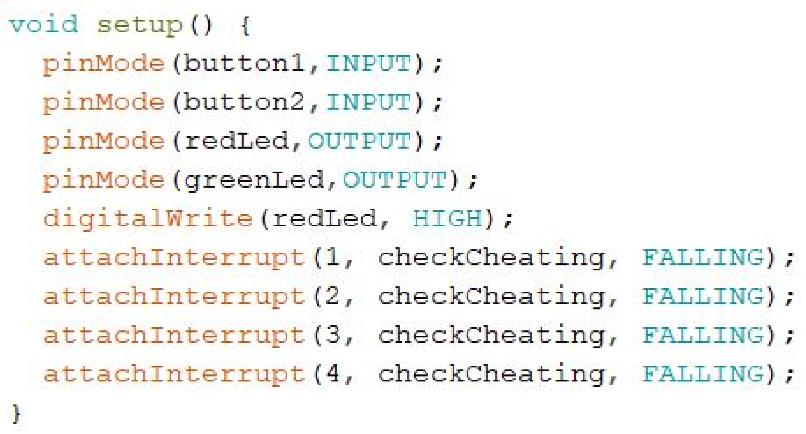
```
const int button1 = 1;
const int button2 = 2;
const int button3 = 3;
const int button4 = 4;
const int redLed = 9;
const int greenLed = 8;
const int timeConstant = 2500;      //Change to higher number tc
int lastEntry = 0;                  //The time the user insertec
boolean prevResult;                 //Check if the user insertec
boolean sameButton;                 //Check if the current digit
boolean previResult;                //Check if the user insertec


boolean fail;
int greenLedState = 0;
int redLedState = 1;
int buttonPress = 0;
```

- Boolean: holds one of two values, true or false. (Each bool variable occupies one byte of memory.) It is used to identify a conditional statement, allowing the processor to make a “decision” (take action) if the conditions are met or not.
    - prevResult: Check if the user inserted the previous pin correctly
    - sameButton; Check if the current digit and the next digit of the code is the same
    - previResult:Check if the user inserted the previous pin correctly
    - Fail: “boolean fail” is now identifying the conditions for “false” or when the primary conditions of “true” are not met.
        - greenLedState= 0: When the statement is false it equals 0.
        - redLedState= 1: When the statement is true it’ll be 1.
        - buttonPress= 0: Pressing the button equals an automatic false.

<u>Void setup():</u>

- pinMode (): Configures the specified pin to behave either as an input or an output.
    - (button1, INPUT): Means that when button1 is pushed the data emitted is collected as input.
    - (redLED, OUTPUT): Means that when the future conditions are met the



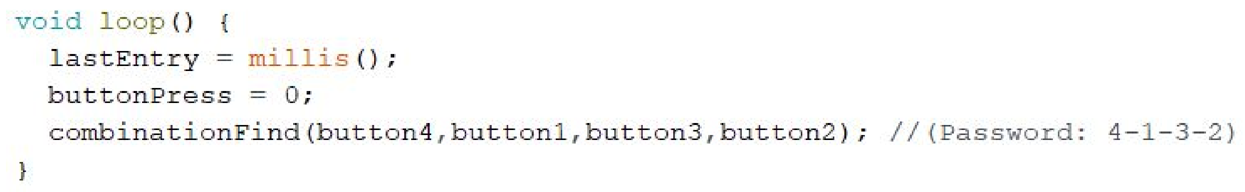
```
void setup() {
  pinMode(button1,INPUT);
  pinMode(button2,INPUT);
  pinMode(redLed,OUTPUT);
  pinMode(greenLed,OUTPUT);
  digitalWrite(redLed, HIGH);
  attachInterrupt(1, checkCheating, FALLING);
  attachInterrupt(2, checkCheating, FALLING);
  attachInterrupt(3, checkCheating, FALLING);
  attachInterrupt(4, checkCheating, FALLING);
}
```

input will trigger an output response, lighting the LED.

- attatchinterrupt(): Interrupts are useful for making things happen automatically in microcontroller programs, and can help solve timing problems.
    - Direct use of interrupt numbers may seem simple, but it can cause compatibility trouble when your sketch is run on a different board.
    - (10, checkCheating, FALLING);: I'm not sure what it is doing exactly but it is checking pin 10 for "cheating".
        - FALLING for when the pin goes from high to low.
        - LOW to trigger the interrupt whenever the pin is low,
        - HIGH to trigger the interrupt whenever the pin is high

<u>Void loop():</u>

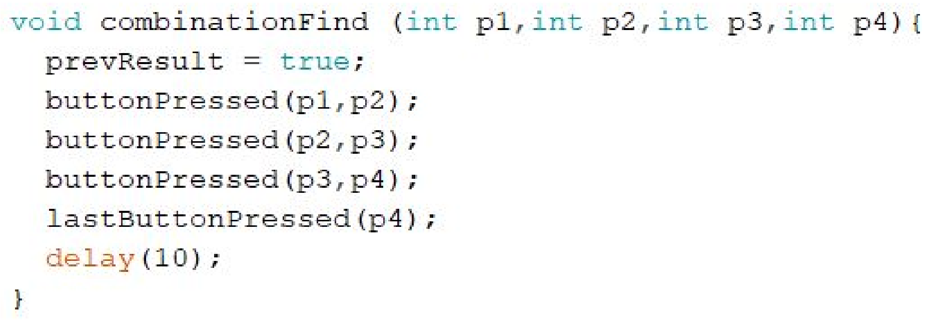
- lastEntry = millis(); When the last button in the combo is pushed the number of seconds since the program started is returned/presented in the data.
    - millis(): Number of milliseconds since the program started (unsigned long)
- combinationFind(button4,button1,button3,button2); //(Password: 4-1-3-2): It is stating the combination to the lock.



```
void loop() {
  lastEntry = millis();
  buttonPress = 0;
  combinationFind(button4,button1,button3,button2); //(Password: 4-1-3-2)
}
```

<u>Void combinationFind (int p1,int p2,int p3,int p4):</u>

- prevResult = true;: If the previous button is pushed correctly the statement is true (maybe red button flash).
- buttonPressed (p1, p2);: This is identifying the sequence order to which the buttons need to be pressed.
- lastButtonPressed(p4);: This is the last button that needs to be pressed.
- delay(10);: 10 millisecond delay before computation and LED light.



```
void combinationFind (int p1,int p2,int p3,int p4){
  prevResult = true;
  buttonPressed(p1,p2);
  buttonPressed(p2,p3);
  buttonPressed(p3,p4);
  lastButtonPressed(p4);
  delay(10);
}
```

<u>Void buttonPressed:</u>

- void buttonPressed (int b1,int b2){: This is the beginning of the Boolean functions that will show the processor how to identify whether the correct combination has been inputted by comparing it to the previous.
- If (b1 == b2){
          sameButton = true;:
  Find out if the current button and the next button are the same, as in they are both true (correct) pushes.
- Else{
          sameButton = false;:
  Write sameButton as false (the current digit and next digit are not the same)
- if (prevResult == true){
          while(millis() - lastEntry < timeConstant && prevResult == true){
  while 2.5 seconds haven't passed and the previous digit entered was correct
- if(sameButton == true){
  //If the two buttons are the same run this code
- if(digitalRead(b1) == HIGH){
  //If the correct button is pressed
- while(digitalRead(b1) != LOW){
  Only move on to the next line when the current button is let go of, so that holding down the button won't register for the next digit of the code
  - digitalRead: Reads the value from a specified digital pin, either HIGH or LOW.
- prevResult = false;
  previResult = false;
          delay(10);
  If the previous result is incorrect the Arduino will return “false.”
  Honestly, I’m not sure why there are two variables that mean the same thing, I feel it makes the code more convoluted, but I’m not expert.
- Break;
  break is used to exit from a for, while or do…while loop, bypassing the normal loop condition. It is also used to exit from a switch case statement.
- This series is repeated for each digit necessary to unlock the combination.
  - Lastbuttonpress is similar in layout to the previous buttons, just with a different variable name.
  - if(buttonPress>3){   //If a button is pressed more than 3 times then restart the code
-