Exercises 9-11 (Amal)

Exercise 9

9a: If the reference voltage is 5 V, then $V_{A0} = \frac{\text{analogRead(A0)}}{1023} \cdot 5$.

9b: First c is declared as a `char`. For each run through of the loop, c is updated. For $i = 0$: $c =' 0'$.
For $i = 1$: $c =' 0' + 1 \cdot 2$, but since arithmetic is done in `int`: $48 + 2 = 50$. 50 encodes the character 2.
For each run of the first statement in the loop, $c =' 0',' 2',' 4',' 6'$.
For each run of the second statement in the loop, the character $^\circ$ is printed to serial.
All in all, we expect the following output: $0^\circ 2^\circ 4^\circ 6^\circ$
But when we actually go to run this code, we see a ? symbol where the degree symbol should have been. This is apparently because the arduino IDE is unicode-based (UTF8). Looking at this webpage https://www.utf8-chartable.de/ we see that two bytes are actually needed to encode the degree symbol: 0x00b0, so this is why we get the ? symbol. Arduino does not know that we want 176 to be 2 bytes long. This could be fixed by using this code instead:

```
char c;
for (int i = 0; i < 4; i++) {
  c = '0' + i * 2;   // <-- FIXED quotes
  Serial.print(c);
  // Serial.print(176);
  Serial.write(0xC2);
  Serial.write(0xB0);
}
```

9c: `print()` converts data into ASCII/UTF-8 text before sending it over the serial, while `write()` transmits the 8-bit (1-byte) number that we have entered in the argument field. However, the Arduino serial monitor will make sense of these binary values by converting them to unicode characters.

9d-9e: *refer to sketch ex9*. I used the fact that according to data sheet for TMP36 (my sensor), scaling factor is 10 mV per degree Celsius and offset is 500 mV at 0 degrees Celcius.

9f: Yes, blowing decreases the temperature and holding the sensor in between my finger increases the temperature.

Exercise 10

10a: Using a glass w/ice cube and glass with tea
Red = above 24 degrees Celsius
Yellow = between 24 and 23 (excluding 23).
Green = below 23

10b: *refer to ex10*

Exercise 11

11a: I2C stands for inter-integrated circuit and is a serial communication protocol that allows the Arduino to communicate with the LCD using only two data lines, SDA and SCL. By using an I2C interface, the LCD requires fewer pins and can share the same bus with other devices, simplifying wiring and also reducing how complex that hardware is.

11b-c: I do not have an LCD screen with I2C, only one without I2C, so I will do this exercise once I have it (I got confirmation that I will get the RFID kit tomorrow otherwise I will do it in class tomorrow when I borrow an LCD screen w/I2C)