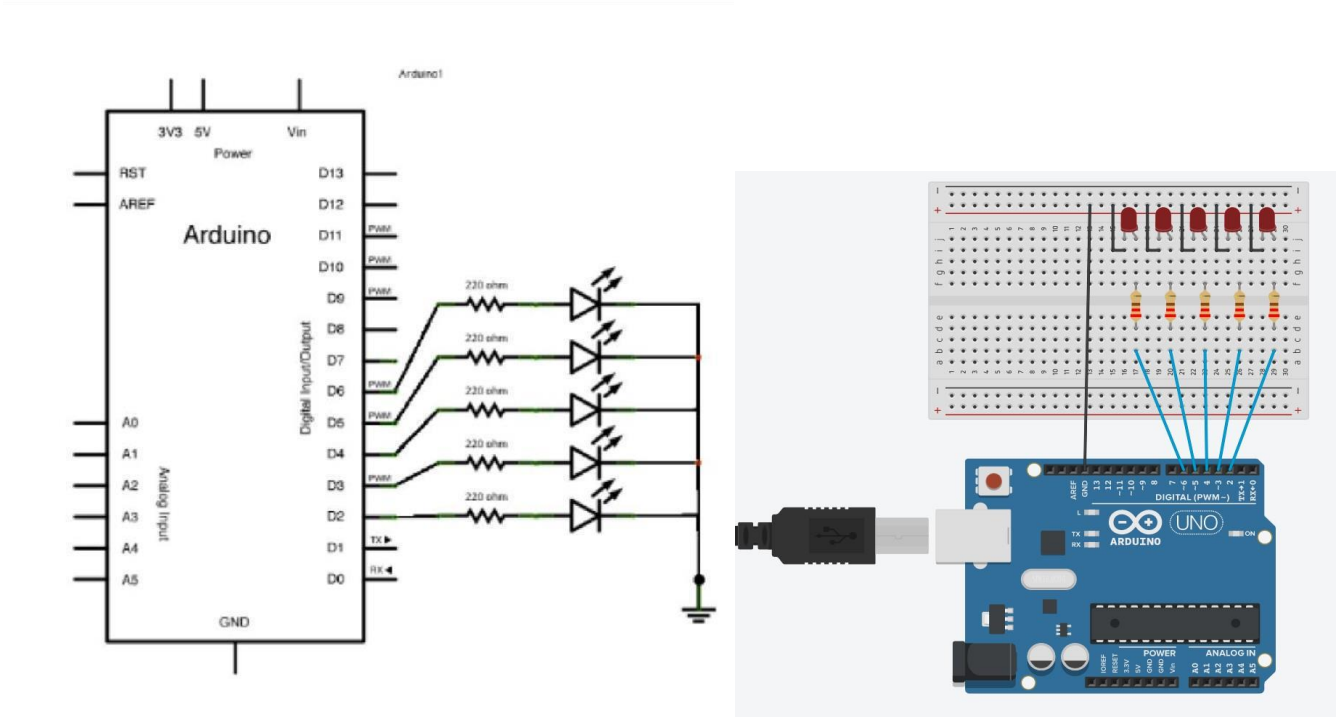
Program 2: Switch Statement

1. Navigate to TinkerCAD > Create the following circuit > Circuit > Create a New Circuit > Rename to Multiple LEDs > Create the following Circuit **Circuit Design**



NOTE: LED Anode (Long Leg) should always be linked to the resistor and Cathode (short Side) linked to the ground or the direction the circuit is going.

Long Leg = Electricity In, + (Positive)

Short Leg = Electricity Out, - (Negative)

Switch statement is an If/Else/If/Else Statement. It only allows the user to compare discrete values (one thing at a time). It is not possible to compare ranges of objects. (For that an If statement is needed).

Using a Switch/Case statement is the equivalent of writing multiple If/Else Statements

For Example

```
switch (inByte)
{
case ‘a’: Serial.println (‘A’);
          break;
case ‘b’: Serial.println (‘B’);
          break;
}
```

If Statement Comparison

```
If (inByte == ‘a’)
{ Serial.println (‘A’);}
else
{ Serial.println (‘B’);}
```

NOTE: break; will break out of the Case Statement, no other conditions have to be checked. See Reference Guide for more info

Program Code

```
/*
  Switch statement with serial input
  Demonstrates the use of a switch statement. The switch statement allows you
  to choose from among a set of discrete values of a variable. It's like a
  series of if statements.
  To see this sketch in action, open the Serial monitor and send any character.
  The characters a, b, c, d, and e, will turn on LEDs. Any other character will
  turn the LEDs off.
  http://www.arduino.cchttps://www.arduino.cc/en/Tutorial/SwitchCase2
*/

void setup() {
  Serial.begin(9600); // initialize serial communication:

  // initialize the LED pins:
  for (int thisPin = 2; thisPin < 7; thisPin++) {
    pinMode(thisPin, OUTPUT);
  }
}

void loop() {
  // read the sensor:
  if (Serial.available() > 0) {

    int inByte = Serial.read();

    // do something different depending on the character received.
    // The switch statement expects single number values for each case; in this
    // example, though, you're using single quotes to tell the controller to get
    // the ASCII value for the character. For example 'a' = 97, 'b' = 98, and so forth:

    switch (inByte) {
      case 'a':
        digitalWrite(2, HIGH);
        break;

      case 'b':
        digitalWrite(3, HIGH);
        break;


      case 'c':
        digitalWrite(4, HIGH);
        break;

      case 'd':
        digitalWrite(5, HIGH);
        break;

      case 'e':
        digitalWrite(6, HIGH);
        break;

      default:
        // Will set code back to a defined state (i.e all LED's on or all LED's OFF)
      break;

    }

  }
}
```

Run Simulation the program to test it.
Notice how when one key is pressed then another the light of the previous key does not turn off.

Assignment: LED Lights Modify the program to do the following

1. Only have the inputted LED Lamp Turned ON (All other LED's turn off with each new input) I.E When 'a' is pressed 'a' LED light is on, then when 'b' is pressed the 'b' LED light is on and the 'a' LED is turned off.
2. Output if Input is NOT a, b, c, d, or e
    a. Blink all of the LED Lamps for a duration of 2 seconds
    b. Output in the Serial Monitor "ERROR, INPUT INVALID!"
    (HINT: if inbyte < 97 || inbyte >101)

NOTE: Serial.read converts all input to ASCCII code form. For example, if lower case z is inputted then the CPU sees a value of 122. ASCCII codes and alpha characters can be used interchangeability.
See below of values

DEC Value	Character
32	space
33	!
34	"
35	#
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

DEC Value	Character
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	C
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93	]
94	^
95	_

DEC Value	Character
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	|
125	}
126	~
127	

Serial Monitor Line Types: The Serial Monitor will assign a value for each line, even if there is no input or when the user is typing in a value to be read by the program.

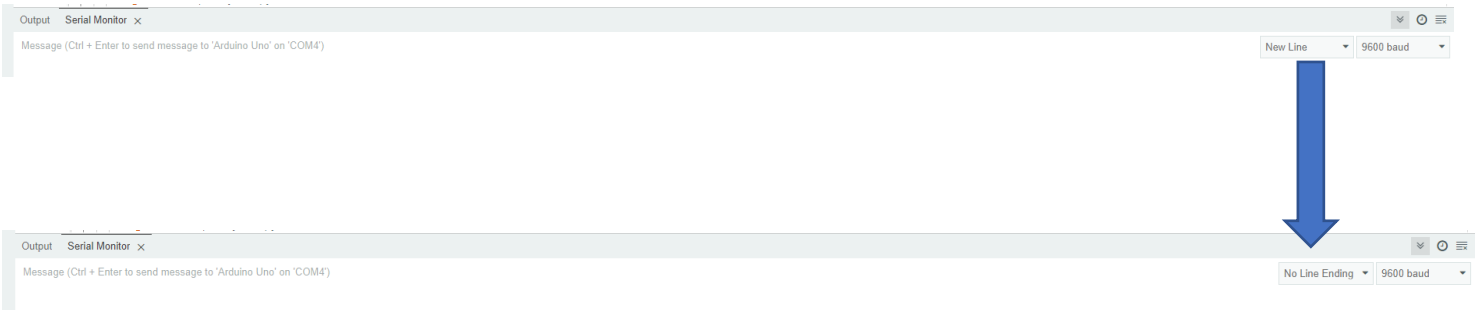
New Line: Inputs a Default Value for the Serial Monitor

No Line Ending: Will Cancel the Default Value for the Serial Monitor

Carriage Return: Used for Non Alpha-Numeric Characters

Both NL & CR: Used are defined in ASCII Code and can send Decimal Values

****Change Line Mode from New Line to No Line Ending****



Submission: Create a Video or Share from Goolge Drive of the Arduino working >
Email To: jourdem@brightonk12.com