JavaScript Lab 3

Upload in your GitHub account and submit your link.
Due Date: Tuesday, 13th

Links that might be helpful:

- Array Methods
- Sorting Arrays
- MDN

Q1

Evaluate JavaScript code fragments **(9 Points)**

a. Given the following array: **(1 Point)**

```
var arr = [ 'dog',  'cat', 'deer'];

var result = arr[0]+arr[2];
```

What is the value of **result**?

b. Given the following program, add the code necessary to assign **children** the concatenation of the arrays **girls** and **boys**. **(2 Points)**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
var girls = ["Cecilie", "Lone"];
var boys = ["Emil", "Tobias", "Linus"];
var children = // add code here
document.getElementById("demo").innerHTML = children;
</script>

</body>
</html>
```

c. Analyze the following code fragment, and answer the questions below. **(2 Points)**

```
var arr = [ 20, 30 ];
for (var i = arr.length; i < 5; i += 1){
   arr[i] = Math.pow(i,2);
}
```

Will the above for loop execute without error?

If it does execute, what will be contained in the array **arr** after the loop terminates?

d. Analyze the following code fragment, and answer the questions below. **(2 Points)**

```
var arr = [ 10, 20, 30, 40, 50, 60, 70, 90 ];
var sum = 0;

for (var i = 1; i < 7; i += 1){
   sum = sum + arr[i];
}
```

What is the value **sum** when the loop terminates?

The sum after the loop terminates should be 370. Fix the necessary statements to ensure sum is 370.

e. Given the following code fragment, what is the output to the console after the completion of this code fragment? **(2 Points)**

```
// code fragment from inside main - start trace here
var DL = 5;
var d = [25.0, 9.0, 10.0, 25.0, 15.0];
var mi = 0;
var m = d[mi];
for (var i = 1; i < DL; i++) {
  if (d[i] < m) {
    mi = i;
    m = d[mi];
    console.log(m);
  }
}
console.log("mi = ",mi," m = ",m);
// code fragment from main .. end trace here
```

Q2.

Write JavaScript code fragment using arrays **(10 Points)**

Given the following spreadsheet, write a function (called sumArray) that will receive an array, and a number representing the size of the array, and return the value of the sum indicated in yellow below.

	A	B	C	D	E	F
1	1	2	3	4	5	15
2	2	3	4	5	6	20
3	3	4	5	6	7	25
4	4	5	6	7	8	30
5	5	6	7	8	9	35
6	15	20	25	30	35	125

Q3.

Create a webapp using HTML/JavaScript **(150 Points)**

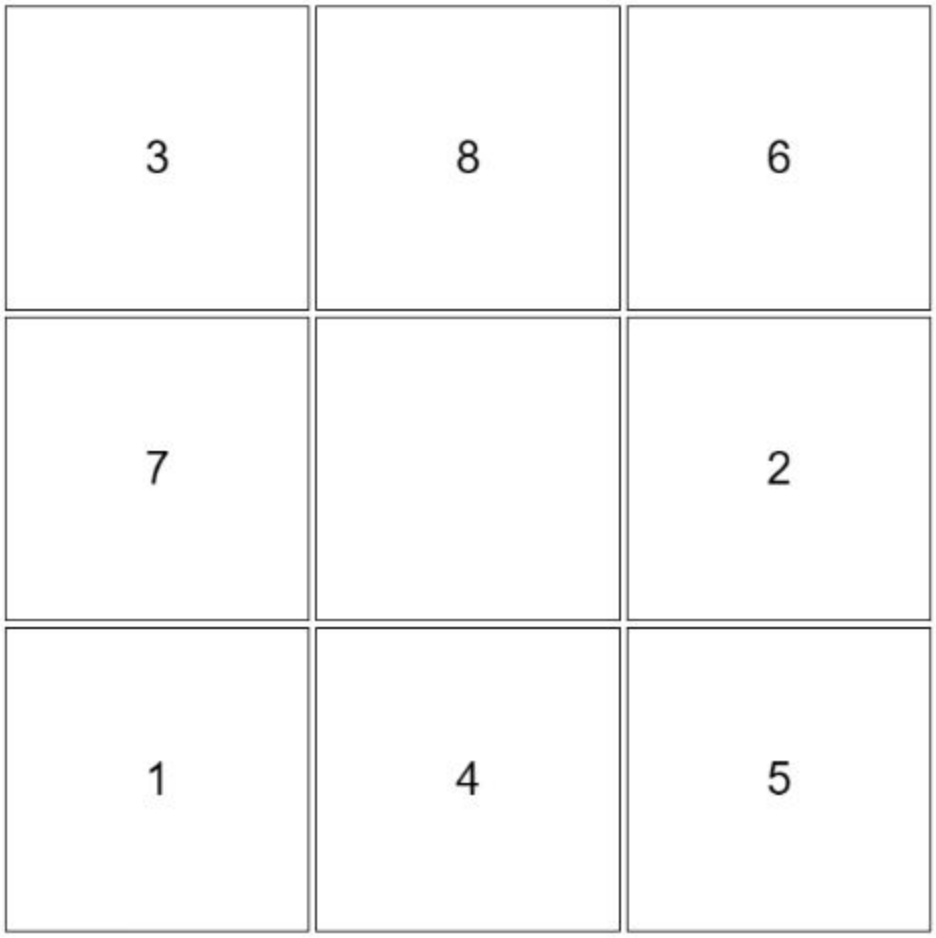
Write a webapp to implement a sliding puzzle. Here is the definition from Wikipedia: "A sliding puzzle, sliding block puzzle, or sliding tile puzzle is a tour puzzle that challenges a player to slide (frequently flat) pieces along certain routes (usually on a board) to establish a certain end-configuration. The pieces to be moved may consist of simple shapes, or they may be imprinted with colors, patterns, sections of a larger picture (like a jigsaw puzzle), numbers, or letters."

Complete the following requirements for this webapp:

a. Create the board and use numbers as the pieces to be moved. **(5 Points)**
b. Use array(s) to store the values of the pieces and/or to store valid moves of pieces. You must implement at least one array. **(30 Points)**
c. Add a "moves" <p> tag to update moves made. **(5 Points)**
d. Add a "Start Time" <p> tag that will show the current hours:minutes:seconds when the webapp is first launched. **(5 Points)**
e. Add a "Restart Game" <button> tag that will restart the game when pressed (no warning necessary). **(5 Points)**

Your webapp should start up similar to the following screenshot:

Sliding Puzzle

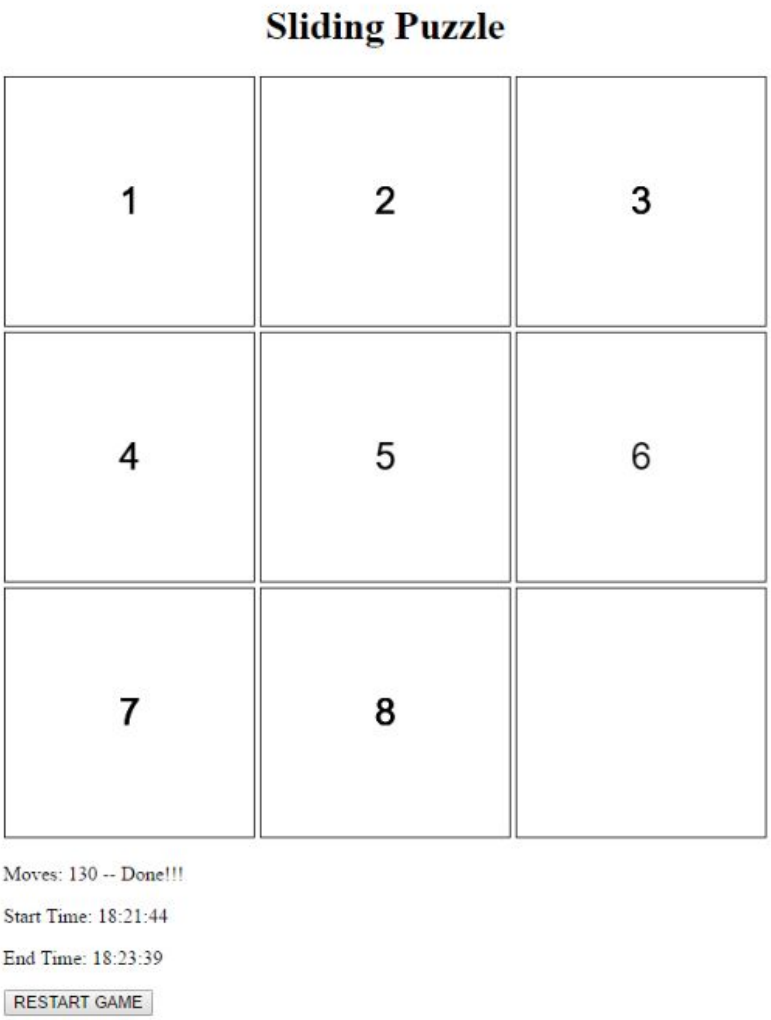


Moves: 0

Start Time: 16:9:18

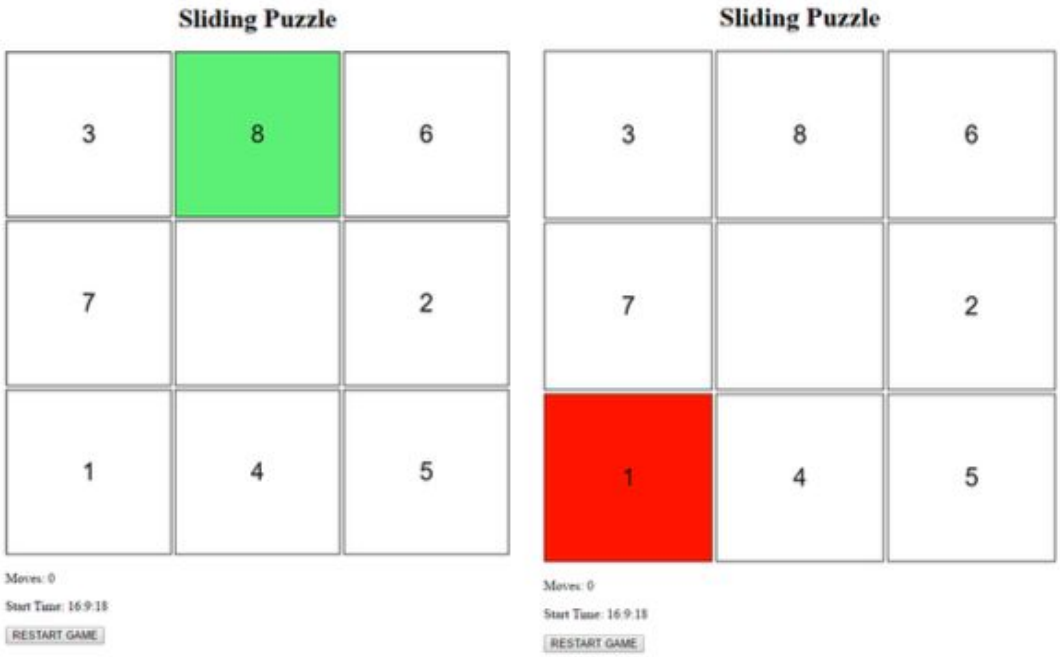
RESTART GAME

f. Add an "End Time" <p> tag that will show the end time when the puzzle is solved. This tag is right below the "Start Time" <p> tag; but does not appear until the puzzle is solved. See example below. **(5 Points)**

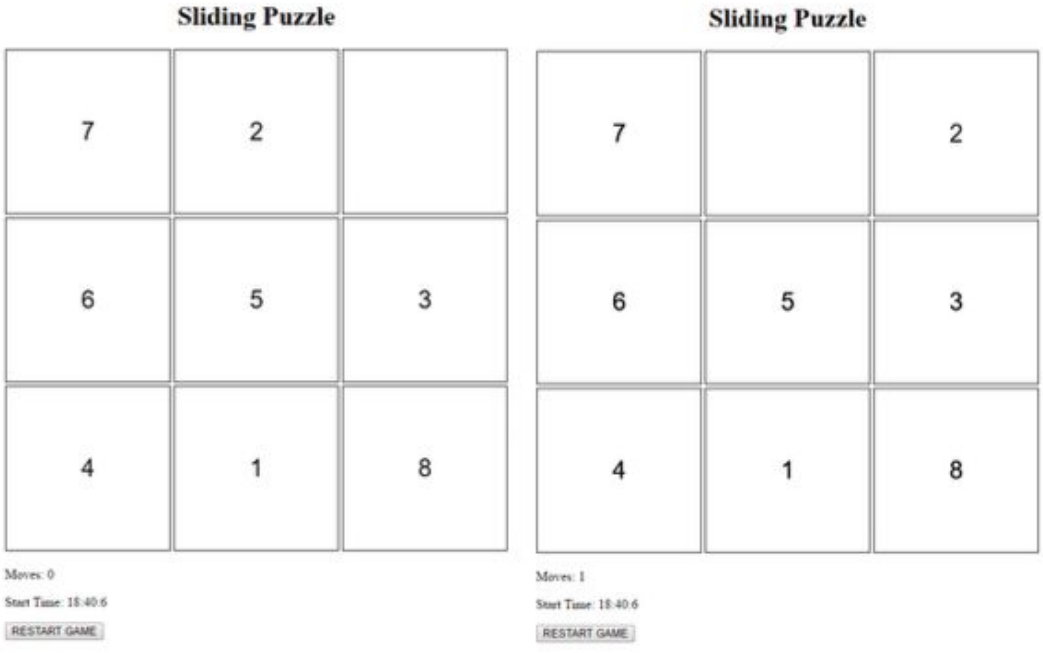


g. Write event handlers for the following:

- "Restart Button" **(15 Points)**
- Change the color of a piece to red if the mouse is over a piece that cannot be moved – reset the color back when the mouse moves away from that piece – see right screenshot below **(25 Points)**
- Change the color of a piece to green if the mouse is over a piece that can be moved – reset the color back when the mouse moves away from that piece – see left screenshot below **(25 Points)**



- Process valid moves (clicks) to a piece – left screenshot below is start of puzzle; right screenshot below is the result of the first piece moved (2 from top-row-center to top-right) **(30 Points)**



Note – your board of pieces may be individual canvases or table cells.