Grading Rubric –
Tutorial 11, Review

Description	Pts	Your Score
1. Use your editor to open the jpf_hitori_txt.html and jpf_hitori_txt.js files from the RA11 folder. Enter your name and the date in the comment section of each file, and save them as jpf_hitori.html and jpf_hitori.js respectively.	**4**	
2. Go to the jpf_hitori.html file in your editor. Directly above the closing </head> tag, link the page to the jpfjiitori.css style sheet and to the jpf_grids3.js and jpfjiitori.js JavaScript files. Load both JavaScript files asynchronously. Take some time to study the contents of the HTML file and then close it, saving your changes.	**4**	
3. Go to the jpfjiiton.js file in your editor. Directly below the comment section, declare the global allcells variable, which you will use to store an array of the puzzle cells in the Hitori table. Do not define a value for the variable yet.	**4**	
4. Insert a command to run the startUp() function when the page is loaded by the browser.	**4**	
5. Add the startUp() function, which displays the contents of Puzzle 1 after the page is loaded and sets up the initial event handlers. Within the function, add the following commands: a. Change the inner HTML of the element with the ID, "puzzleTitle" to the text "Puzzle 1". b. Call the drawHitori() function using the hitorilNumbers, hitorilBlocks, and hitorilRating variables as parameter values and store the HTML code returned by the function in the inner HTML of the page element with the ID "puzzle". c. Declare a variable named puzzleButtons referencing the page elements with the class name "puzzles". Loop through the puzzleButtons object collection and for each button add an event handler that runs the switchPuzzle() function when the button is clicked. d. Call the setupPuzzle() function that defines the initial appearance of the first puzzle. e. Add an event handler to the Check Solutions button to run the findErrors() function when clicked. f. Add an event handler to the Show Solutions button to run the showSolution() function when clicked.	**12**	

6. Add the switchPuzzle() function, which switches the page between the three possible Hitori puzzles. Include the event object e as a parameter of the function and add the following commands: a. Declare the puzzlelD variable equal to the ID of the event object target. b. Change the inner HTML of the element with the ID "puzzleTitle" to the value of the value attribute of the event object target. c. Create a switch-case structure with the puzzleID variable that loads the appropriate HTML code for each of the three puzzles into the page element with the ID "puzzle". Use the drawHitori() function to generate the HTML code and assume that puzzleID is limited to the values "puzzlel", "puzzle2", and "puzzle3". d. After the switch-case structure, call the setupPuzzle() function to set up the features of the selected puzzle. e. Enclose all of the commands in the switchPuzzle() function within an if statement that displays a confirm dialog box asking users whether they want to switch puzzles even though their work will be lost. If the confirm dialog box returns a value of true, run the commands within the if statement command block.	**20**	
7. Create the setupPuzzle() function to set up the features of the puzzle table. Within the function add the following commands: a. Use the querySelectorAll() method to create an object collection of all of the td elements within the hitoriGrid table and save the object collection in the allCells variable. b. Create a for loop that loops the allCells object collection and, for each cell, change the background-color style to white, the font color to black, and the border-radius value to 0. c. Within the for loop, add a mousedown event listener for each cell in the allCells collection that changes the cell's appearance depending on whether the Shift key, the Alt key, or no key is pressed by the user. Add the following commands to the anonymous function for the mousedown event: i. Change the background color to white, the font color to black, and the border radius to 0 if the user is pressing the Shift key. ii. Change the background color to black, the font color to white, and the border radius to 0 if the user is pressing the Alt key. iii. Otherwise, change the background color to rgb(101, 101, 101), the font color to white, and the border radius to 50%. iv. To avoid inadvertently selecting the text of the table cells, include a command to prevent the default action of the browser in response to the mousedown event. d. Rebecca wants a different mouse cursor depending on whether the user is pressing the Shift key, the Alt key, or no key when the mouse pointer moves over a puzzle cell. Within the for loop, add a mouseover event listener for each puzzle cell that runs an anonymous function that i. Changes the cursor to the jpf_eraser.png image or the generic cursor named "alias" if the user is pressing the Shift key.	**34**	

ii. Changes the cursor to the jpf_block.png image or the generic cursor named "cell" if the user is pressing the Alt key. iii. Otherwise, changes the cursor to the jpf_circle.png image or the generic cursor named "pointer". e. Finally, within the for loop, add an event listener that runs the checkSolution() function in response to the mouseup event to test whether the user has solved the puzzle.		
8. Create the findErrors() function that will highlight incorrect cells by displaying the cell number of an incorrect cell in a red font. Add the following commands: a. Create a for loop that goes through all of the cells in the allCells object collection. If the cell belongs to the blocks class but has a background color of rgb(101, 101, 100) or if it belongs to the circles class but has a black background, change the font color to red. b. The red font colors should appear only briefly. After the for loop, insert a setTimeout() method with a 1-second interval. Within the setTimeout() method, add an anonymous func- tion that loops through every cell in the allCells collection, changing all cells with a font color of red back to white.	**8**	
9. Document your code in the JavaScript file with descriptive comments throughout.	**8**	
10. Save your changes to the file and then load jpf_hitori.html in your browser. a. Verify that you can switch puzzles by clicking the Puzzle buttons at the top of the page, and that you are prompted to confirm whether you want to change your puzzle. Verify that you can view the complete solution to each puzzle by clicking the Show Solution button. b. Verify than you can change a cell to a gray circle by clicking the cell. Verify that you can change a cell to a solid black block by clicking the cell with the Alt key pressed down. Finally, verify that you can restore a cell to black text on a white background by clicking a previously selected cell with the Shift key pressed down. c. Verify that the cursor changes shape as you move the mouse pointer over the puzzle cells, changing from a circular cursor to a block cursor when the Alt key is pressed or to an eraser cursor when the Shift key is pressed. d. Verify that you can test for errors by clicking the Check Solution button, and that your errors are displayed in a red font for one second. e. Solve the first puzzle using the solution provided in Figure 11-51. Verify that you receive a congratulatory message upon successfully completing the puzzle.	**2**	
TOTAL	**100**	

YOUR SCORE: _____