M319: AP Computer Science Principles Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Chapter 3: Arrays

Monster Match Project Date: \_\_\_\_\_\_ Teacher: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Monster Match Project**

**Instructions:** You will work individually to create a website to play a game called “Monster Match”

**Submission Guidelines:** You must upload each of the following:

* A .pdf printout of your source code
* A 1-minute (or less) screencast of you showing the functionality of your website. This should show off all major features of your program.

**Requirements:**

1. Standard HTML and CSS elements (a title, basic CSS styling, a header)
2. Descriptive code comments (including documentation, code citations for functions)
3. A set of 4 monster images, along with proper image citations
4. Global variables inside <script> tag:
   1. A variable representing Player 1’s deck of cards. You do not need to initialize this now.
   2. A variable representing Player 2’s deck of cards. You do not need to initialize this now.
   3. A variable representing the currently-selected index from Player 1’s deck. You do not need to initialize this now.
   4. A variable representing the currently-selected index from Player 2’s deck. You do not need to initialize this now.
   5. A variable representing the current card from Player 1’s deck. You do not need to initialize this now.
   6. A variable representing the current card from Player 2’s deck. You do not need to initialize this now.
   7. A variable representing the number of cards per deck. Initialize this to the number 8.
   8. A variable representing an array of file names for the monster cards. This should be initialized with the four different file names from the Monster Pictures folder.
   9. A string variable containing the current game state. This should be initialized to the string “Game Over”.
5. A button to start a new game. This should call an appropriate function when clicked.
6. A button to view game instructions. This should call an appropriate function when clicked.
7. The following HTML <table>:



1. A <div> element that allows you to display important messages about the game.

Ex: “The monsters have matched, good catch!”, “Uh oh, no match this time :(“

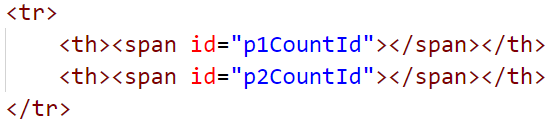
1. A <div> element that allows you to display the instructions about how to play the game when the user has clicked the 'Show Instructions' button.

**B-Level Requirements:**

1. Add a global variable to keep track of your personal score. This should be initialized to an appropriate starting value.
2. Add a <div> tag with an appropriate ID that will display your current score.
3. A function with two parameters that returns a random integer between the two values given (inclusive). *Hint: Look at your Marvel Trivia website.*
   * Make sure to give proper citation credit to the author when necessary!
4. A function that takes two parameters (an array of filenames and an index) that returns a formatted string in the form of an <img> tag. *Hint: Look at your Marvel website.*
5. A function that shows the game instructions when the user clicks the instructions button. These should be displayed in the appropriate <div> tag.
6. A function that creates a string representing your current score, and displays it in the appropriate <div> tag. This should utilize your score variable.
   * Ex: “Score: 5”, “Score: 9”
7. A function that “deals” a random card from each player’s deck. Generate a new random number for each player that represents a valid index in of their 8-card deck. Then, store the name of that image stored at that index. Format a string in the form of an <img> tag with the chosen image name for each player (is there a function that does this?) and insert them into their appropriate image ID’s. Set the game state to “Play”, and display a message with general rules in the appropriate <div> tag.
   * Ex: “Press ‘Y’ if matching or ‘N’ if not matching”
8. A function that starts a new game of Monster Match. This should set each player’s deck to empty arrays, then use a for-loop that puts eight random monster image names in each player’s deck. It should insert “Player 1” and “Player 2” in each player’s ID (in the <table> tag), deal one new card for each player (is there a function that does this?), and display the current score (is there a function that does this?)
9. A function that checks for a win. If you earn a score of 10, display an appropriate winning message on the website and set the game state to “game over”.
   * Ex: “Congratulations, you’ve won! Press ‘Start New Game’ to play again”.
10. A function named “checkMatch(event)” that controls the main logic of the game.
    * When this function is called, use a variable to store the key pressed from the event.
      + *Hint: event.key - produces the string version of the key pressed.*
    * If the state of the game is “play”:
      + If the key “y” was pressed, check if the cards match.
        - If they match, the user got it correct! Display a congratulatory message that includes instructions to press the Space key to deal more cards. Update your score to be one higher than it was.
        - Otherwise the user got it incorrect. Display an appropriate message that includes instructions to press the Space key to deal more cards. Update your score to be one lower than it was.
        - Finally, set the game state to “pause”
      + Otherwise if the “n” key was pressed, check if the cards did not match.
        - Write a similar algorithm to the one you wrote for “y”, but adjust appropriately for this scenario.
      + Otherwise, the user pressed an invalid key.
        - Display an appropriate error message that includes the general rules.
          * Ex: “Invalid key. Press ‘Y’ if matching or ‘N’ if not matching”
    * Otherwise, if the game state is “pause”:
      + If the keypress is a space, set the game state to “Play” and deal one new card for each player (is there a function that does this?)
11. Add the following event handler attribute to your <body> tag:
    * onKeyUp="checkMatch(event)"

**A-Level Requirements:**

1. Add another row to your <table> tag.



1. Delete the code in your function that displays the current score. Replace it with code that instead displays each player’s card total of their deck in their appropriate count ID’s
2. Make the following adjustment to the function that deals out new cards:
   1. Update the output message to match the new rules of the game
3. Make the following adjustment to the function that starts your game:
   1. Instead of giving each player eight *random* cards, give each player exactly two of each monster.
4. Make the following adjustments to the function that checks for wins:
   1. Instead of winning when you reach a score of 10, a player wins when the other player runs out of cards in their deck. Check if either player has no cards in their deck. If so, display an appropriate winning message.
      1. Ex: “Player 1 has won! Press ‘Start New Game’ to play again.”
5. Make the following adjustments to the checkMatch(event) function:
   1. If the game state is currently in “Play” mode.
      1. If the keypress is “a”, check if the cards match.
         1. If they match, push Player 2’s card into Player 1’s deck and then splice Player 2’s card out of their deck. Display a message that says that Player 1 has “claimed the match” and to press “n” to deal the next card.
         2. Otherwise, push Player 1’s card into Player 2’s deck and then splice Player 1’s card out of their deck. Display a message that says that Player 1 has “misfired” and is giving up their card, and to press “n” to deal the next card.
         3. Finally, set the game state to “pause”, call the function that updates counters on the screen, and call the function that checks for wins
      2. Otherwise if the keypress is “l” (a lowercase “L”), check if the cards match.
         1. Then modify write a similar algorithm to the one you wrote for “a”, but adjust as appropriate for “l”
      3. Otherwise if the keypress is “n”, check if the cards match.
         1. If the cards match, display a message informing the players that there is a match on the board and that someone must claim it. Set the game state to “play”.
         2. Otherwise, deal each player a new card (is there a function for that?) and set the game state to “play”
      4. Otherwise, the users pressed an invalid key.
         1. Display an appropriate error message that includes the general rules.
            1. This should include each player’s designated key to press, as well as what key indicates no match.
   2. Otherwise, if the game state is currently in “Pause” mode.
      1. If the keypree is “n”, set the game state to “Play” and deal each player a new card (is there a function that does this?)
   3. Check if either player has won (is there a function that does this?)
   4. Display each user’s score (is there a function that does this?)

**Video Demo:**

*Note: Page layout, CSS, images, etc should match your personal tastes. Don’t just reproduce what you see in the video!*

B-Requirement:

<https://www.youtube.com/watch?v=fYUmVQIXWIw>

A-Requirement:

<https://youtu.be/xpNWCuZFWjw>

**Optional Enhancements:**

1. Customize your CSS styling. You might try using CSS to style your buttons to your liking. There are a lot of great sources for making custom buttons:
   1. <https://www.w3schools.com/css/css3_buttons.asp>
   2. <https://fdossena.com/?p=html5cool/buttons/i.frag>
   3. Just make sure to give code citations if necessary
2. Further customize your monster images. You might be interested in monster images using the .png file format with transparent backgrounds:
   1. Example: <https://pngtree.com/so/monster>
   2. Just make sure to give proper image citations if necessary
3. Allow players to select the size of their deck (by default, each player has a deck of 8 cards, but perhaps they want to play with a deck of 12 cards, 16 cards, or even 40 cards!)
4. Rename your “Show Instructions” button as “Toggle Instructions.” If this button is clicked when the instructions are not visible, they should appear on the webpage. If the button is clicked when the instructions are visible, they should disappear.
5. Keep track of previous matches. Show the user how many games Player 1 and Player 2 have won since the page was loaded.
6. You may add additional enhancements to your game if desired.

**Code of Conduct Reminder:**

While collaboration is always encouraged, plagiarism is never allowed. This is an individual project. Please refer to the Code of Conduct for more information regarding course expectations and consequences for violating these expectations.

**Function Summaries for A-Requirement:**

randomInt(min,max)

* Accepts a minimum and maximum value
* Returns a random integer between [min,max] inclusive

formatImage(inputArray, index)

* Accepts the array of images and a corresponding index
* Creates code for an <img> tag with the appropriate file name inserted as the source

startNewGame()

* Instructions should disappear from the webpage
* Game state should be set to play mode
* Each player’s deck should be refilled with the original eight cards
* Player’s names should appear in their corresponding <span>
* Call to dealNewCards() should be made

updateCounters()

* Number of cards currently in each deck should be display in the appropriate <span> tag

dealNewCards()

* A random index in the range of each player’s deck is generated, and the corresponding image is formatted and displayed in the appropriate <span> tag
* Call to updateCounters() should be made

checkForWins()

* Check each player to see if their deck is currently empty
* If either player’s deck is empty display an according message in the appropriate <div> tag
* Sets the game state to game over

checkMatch(event)

* Accepts an “event” (key pressed) and checks which user pressed their key first
* If cards are matching, player who pressed first takes the other player’s card
* If cards are not matching, player who pressed first gives the other player their card
* If a card is exchanged, calls to updateCounters() and checkForWins() are made
* Deals a new card to the board if no match is present