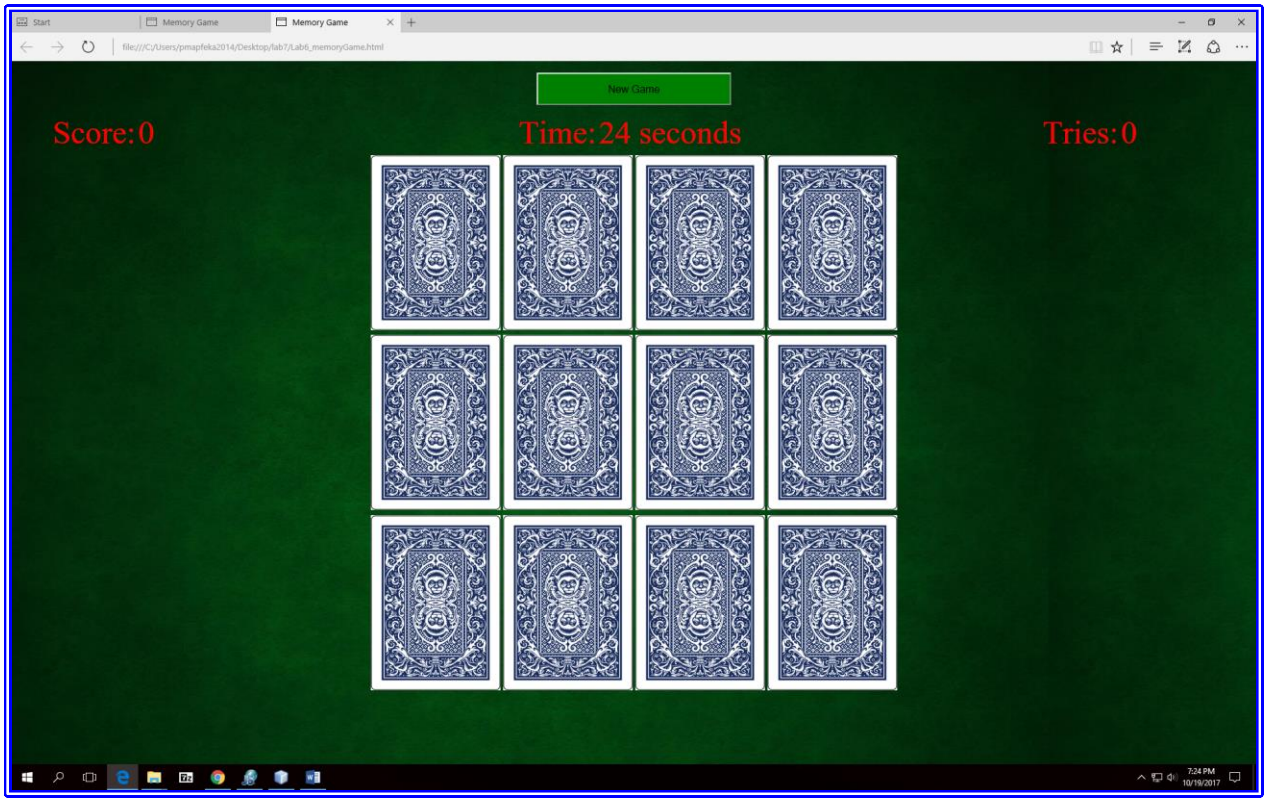
MULTIFARIOUS SYSTEMS 1
ECE 3553

Multifarious Systems Laboratory Report 7
Fall 2017



Student email; pmapfeka2014@my.fit.edu

Lab Partners; none

Date performed; 18 October 2017

Date Submitted; 19 October 2017

Lab #; 7 (Card game in Java script part2)

Lab Instructor; Max Ble

1. INTRODUCTION

This lab continues on the previous lab which was based on designing a memory game. This lab will extend on the previous one by adding timers, random number generators, score tracking and sounds to make the game more interactive.

2. REQUIREMENTS

1. Implement your shuffle function in Lab7_MemoryGame.html
2. Every time the New Game button is clicked, the cards (images) are shuffled.
3. Add a scoring system to your game. Increase the points every time a pair is found.
4. Disable the cards that remain face up. You can do that by disabling the specified tags.
5. Add number of tries. Increment the number of tries every time two cards are shown. Don’t increment tries when the user clicks on the “disabled cards”.
6. Add a countdown timer. Start it when the user clicks on New Game, and set its duration to 20 or 30 seconds.
7. When the user finishes the game, meaning all cards are face up, prompt a JavaScript alert to congratulate him/her. In your alert, show the score, time (duration), tries. Show everything in a single alert.
8. If the timer reaches 0s before all pairs are found, prompt a JavaScript alert showing the score and tries
9. Add a sound effect when the user finds a correct pair
10. Add a sound effect when the user selects an incorrect pair

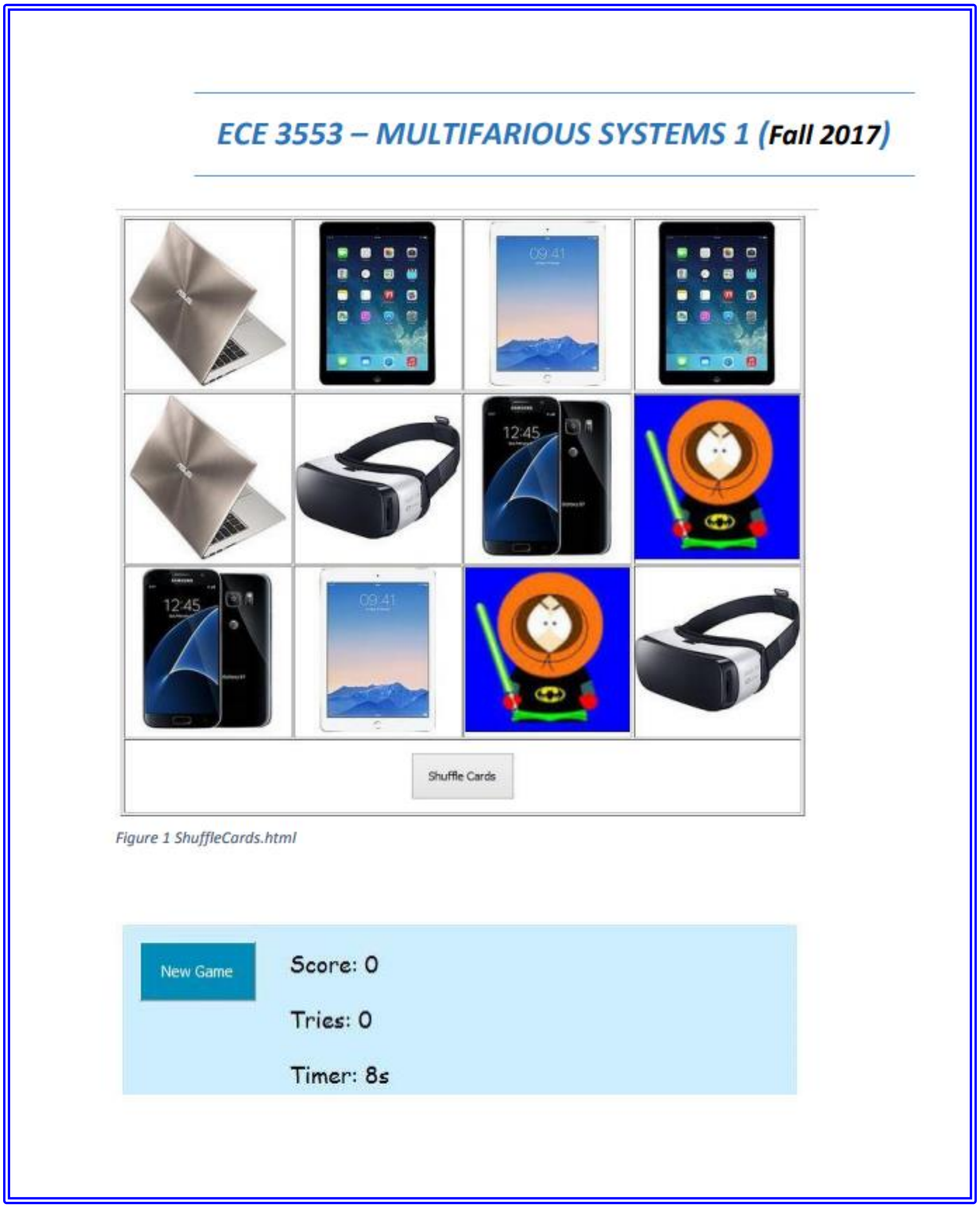

ECE 3553 – MULTIFARIOUS SYSTEMS 1 (Fall 2017)

Figure 1 ShuffleCards.html

Score: 0

Tries: 0

Timer: 8s

Fig1: Expected output

3. METHOD OF SOLUTION
REQUIREMENT #1

Implement your shuffle function in Lab7_MemoryGame.html

□ **SOLUTION:**

To shuffle the cards, a shufflecards() function was created. This function does 3 main things;

- Generate a random number
- Use random number to pick a random card from array and add it to game
- Reorganize array

To begin with, the program has an array of 12 elements, filled with a predetermined order of cards. This is the array we will be picking cards out of. This is similar to picking out of a hat. The image below depicts the array visually;

0	acespade
1	joker
2	king
3	queen
4	5hearts
5	8diamonds
6	acespade
7	joker
8	king
9	queen
10	5hearts
11	8diamonds

We need to be able to generate a random number between 0 and 11 inclusive for the first pick, but it is worth noting that the picking has no replacement, so next time we pick we will only be able to pick from 11 items instead of 12. We will need to generate a random number between 0 and 10 inclusive the next time. Thus the random number generator must generate a random number between 0 and the size of the array.

When we have generated a valid random number, we need to move the card at that number to the array where the cards will be displayed from. For example, if we randomly generated 5, we would need to move 8 of diamonds to the output array. The diagram below shows this;

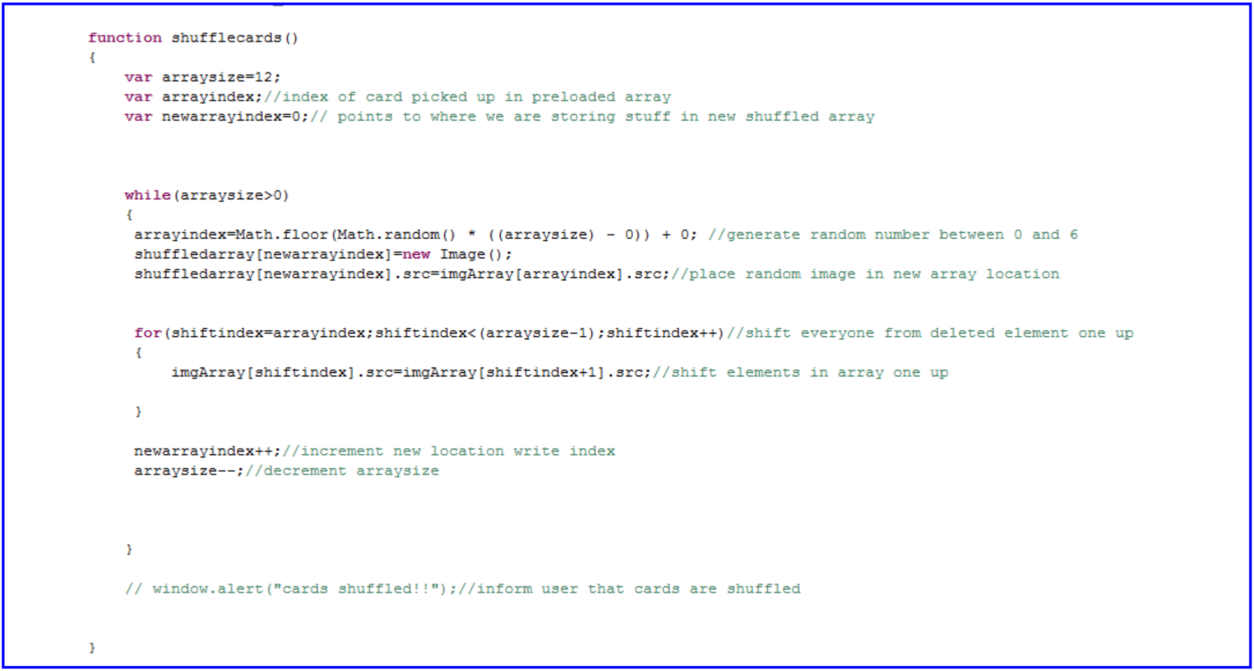
0	acespade
1	joker
2	king
3	queen
4	5hearts
5	8diamonds
6	acespade
7	joker
8	king
9	queen
10	5hearts
11	8diamonds

0	8diamonds
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

After moving the card to the game, we need to delete the card or remove it from our “hat” since there is no replacement. One way to do this is shift everything after the particular cell upward by one. This will not change the size of array, but will give the same effect as deleting the specified cell. We will still have 12 items in our array, but we will have a sizeofarray variable which will be decremented once the shifting of elements occurs. This way the computer will be fooled that the size of the array has gone down. Thus when the random number generator looks up the size of the array on the next random number generation process, it will find it decremented by one. The extra card stored at position 11 will be redundant because as far as the computer knows, the array is not that big. The diagram below shows the result of shifting elements one up starting from element after the one that was picked out to the second last element in the array or arraysize-2.

0	acespade
1	joker
2	king
3	queen
4	5hearts
5	acespade
6	joker
7	king
8	queen
9	5hearts
10	8diamonds
11	8diamonds

The process continues until the output array is filled.



```
function shufflecards()
{
    var arraysize=12;
    var arrayindex;//index of card picked up in preloaded array
    var newarrayindex=0;// points to where we are storing stuff in new shuffled array



    while(arraysize>0)
    {
     arrayindex=Math.floor(Math.random() * ((arraysize) - 0)) + 0; //generate random number between 0 and 6
     shuffledarray[newarrayindex]=new Image();
     shuffledarray[newarrayindex].src=imgArray[arrayindex].src;//place random image in new array location


     for(shiftindex=arrayindex;shiftindex<(arraysize-1);shiftindex++)//shift everyone from deleted element one up
     {
         imgArray[shiftindex].src=imgArray[shiftindex+1].src;//shift elements in array one up

     }

     newarrayindex++;//increment new location write index
     arraysize--;//decrement arraysize



    }

    // window.alert("cards shuffled!!");//inform user that cards are shuffled


}
```

Fig2: shufflecards() function

arraysize	Keeps track of current size of presorted array
arrayindex	Used to store randomly generated number
newarrayindex	Used to store where in shuffled array the new data should be stored
shuffledarray	This array stores the cards that have been randomly picked from presorted array
imgArray	This is the presorted array
shiftindex	This is an indexing variable for the for loop. It is used in the loop for shifting elements one up from their current position.

The math.random() function produces random floats between 0 and 1 but not including 1. Thus multiplying the float by an integer multiples the range by that much. If the size of the array is 3, and we multiply the math.random() function by 3, we get random numbers from 0 to 3. If we add some constant to this value, like 5 per say, we get random numbers from 5 to 8. The equation for this is

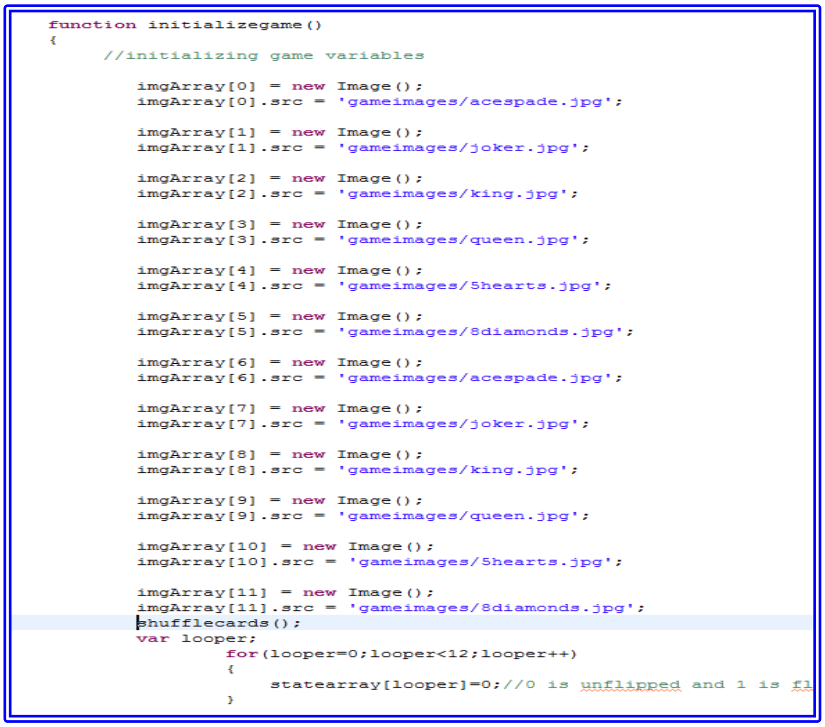
$$y = mx + c$$

Where y is the final generated random number. C is the additive constant and m is the multiplicative constant.The min and max are subject to the formula baove.

REQUIREMENT#2

Every time the New Game button is clicked, the cards (images) are shuffled.

SOLUTION:



```
    function initializegame()
    {
         //initializing game variables

            imgArray[0] = new Image();
            imgArray[0].src = 'gameimages/acespade.jpg';

            imgArray[1] = new Image();
            imgArray[1].src = 'gameimages/joker.jpg';

            imgArray[2] = new Image();
            imgArray[2].src = 'gameimages/king.jpg';

            imgArray[3] = new Image();
            imgArray[3].src = 'gameimages/queen.jpg';

            imgArray[4] = new Image();
            imgArray[4].src = 'gameimages/5hearts.jpg';

            imgArray[5] = new Image();
            imgArray[5].src = 'gameimages/8diamonds.jpg';

            imgArray[6] = new Image();
            imgArray[6].src = 'gameimages/acespade.jpg';

            imgArray[7] = new Image();
            imgArray[7].src = 'gameimages/joker.jpg';

            imgArray[8] = new Image();
            imgArray[8].src = 'gameimages/king.jpg';

            imgArray[9] = new Image();
            imgArray[9].src = 'gameimages/queen.jpg';

            imgArray[10] = new Image();
            imgArray[10].src = 'gameimages/5hearts.jpg';

            imgArray[11] = new Image();
            imgArray[11].src = 'gameimages/8diamonds.jpg';
            shufflecards();
            var looper;
                    for(looper=0;looper<12;looper++)
                    {
                        statearray[looper]=0;//0 is unflipped and 1 is fl
                    }
```

Fig3: shufflecards() function inside initializegame() function

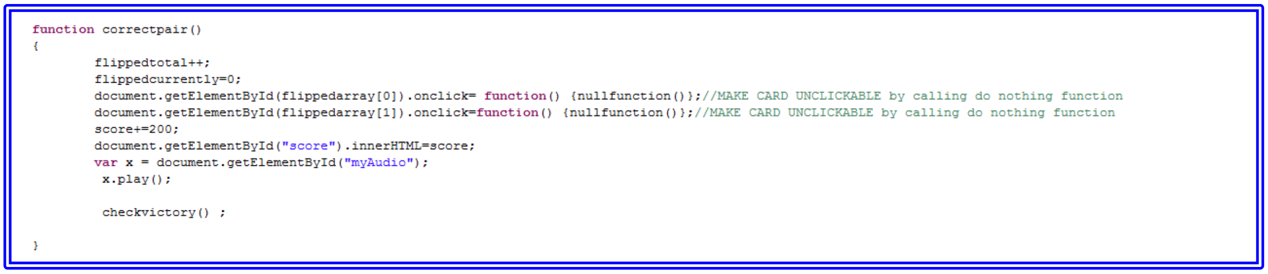
The shufflecards() function is called in the initializegame() function, which in turn is called when the new game button is clicked. Thus the cards are shuffled every time the new game button is clicked

<u>REQUIREMENT#3</u>

Add a scoring system to your game. Increase the points every time a pair is found

SOLUTION:

Upon finding a correct pair, the program will give the user 200 points. So the total number of points is 1200points, since the game has 6 pairs.



```
function correctpair()
{
        flippedtotal++;
        flippedcurrently=0;
        document.getElementById(flippedarray[0]).onclick= function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
        document.getElementById(flippedarray[1]).onclick=function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
        score+=200;
        document.getElementById("score").innerHTML=score;
        var x = document.getElementById("myAudio");
         x.play();

         checkvictory() ;

}
```

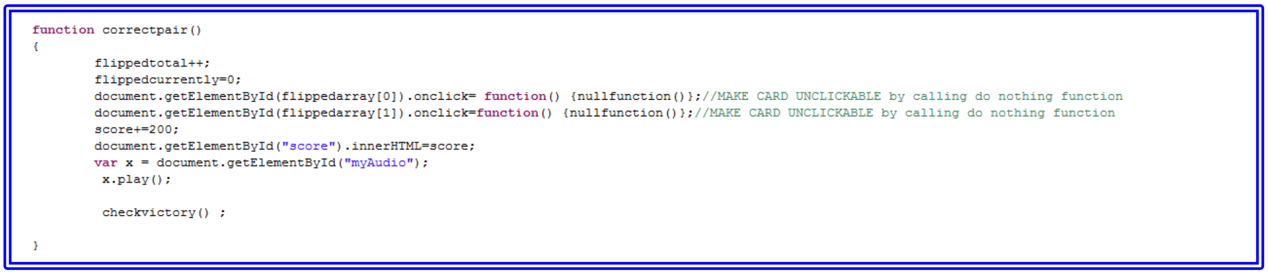
Fig4: correctpair() function

The correct pair function will add 200 points to the player each time they flip a correct pair, since this function only runs when a correct pair is flipped. The score is then updated on the user interface.

<u>Requirement#4</u>

Disable the cards that remain face up. You can do that by disabling the specified tags.

SOLUTION:

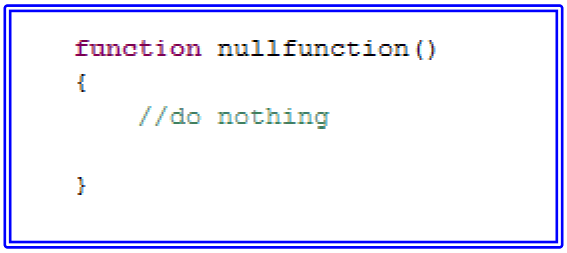


```
function correctpair()
{
        flippedtotal++;
        flippedcurrently=0;
        document.getElementById(flippedarray[0]).onclick= function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
        document.getElementById(flippedarray[1]).onclick=function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
        score+=200;
        document.getElementById("score").innerHTML=score;
        var x = document.getElementById("myAudio");
         x.play();

         checkvictory() ;

}
```

Fig5: disabling flipped up cards



```
function nullfunction()
{
    //do nothing

}
```

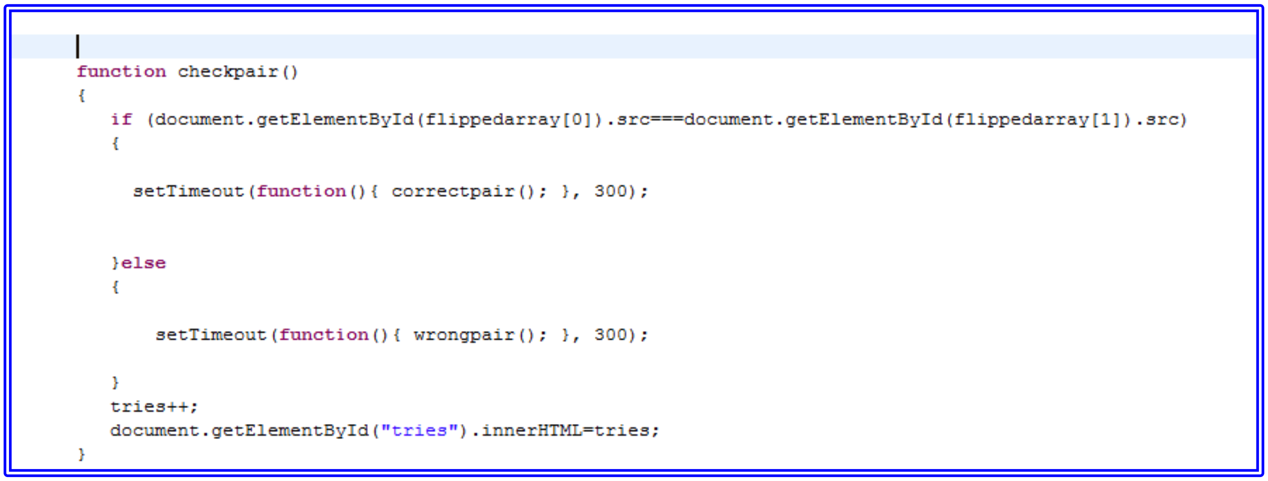
Fig6: nullfunction()

The nullfunction() is a function that does nothing like its name suggests. When a correct pair is flipped, the onclick property of the flippedcards changes to the nullfunction(). This means that when these cards are clicked, they will not flip, they will simply do nothing.

<u>Requirement#5</u>

Add number of tries. Increment the number of tries every time two cards are shown. Don’t increment tries when the user clicks on the “disabled cards”.

SOLUTION:



```

function checkpair()
{
   if (document.getElementById(flippedarray[0]).src===document.getElementById(flippedarray[1]).src)
   {

     setTimeout(function(){ correctpair(); }, 300);


   }else
   {

       setTimeout(function(){ wrongpair(); }, 300);

   }
   tries++;
   document.getElementById("tries").innerHTML=tries;
}
```

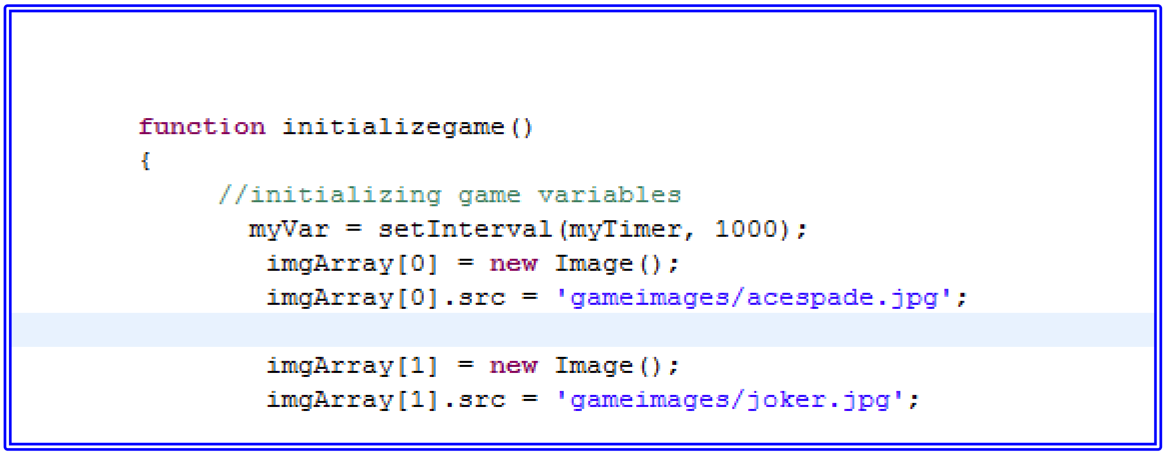
Fig7: checkpair() function

The checkpair function is the function that is run when 2 cards are face up, thus the variable “tries” has been created as a global variable and is incremented when this function runs. The user interface is updated accordingly. Because correctly flipped cards have their onclick event set to nullfunction(), they wont increment tries when flipped.

Requirement#6

Add a countdown timer. Start it when the user clicks on New Game, and set its duration to 20 or 30 seconds.

SOLUTION:

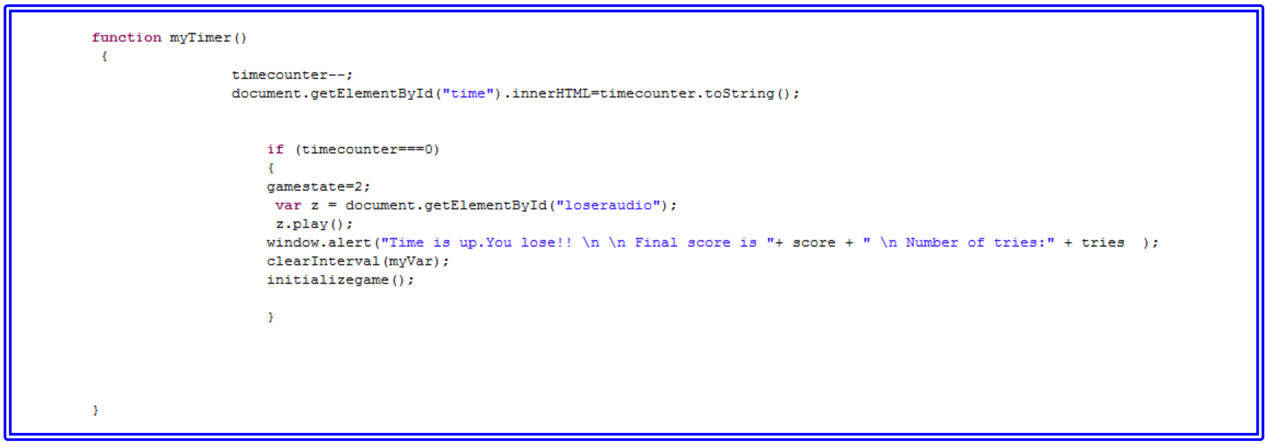


```
function initializegame()
{
      //initializing game variables
        myVar = setInterval(myTimer, 1000);
         imgArray[0] = new Image();
         imgArray[0].src = 'gameimages/acespade.jpg';

         imgArray[1] = new Image();
         imgArray[1].src = 'gameimages/joker.jpg';
```

Fig8: initializegame() function

The setInterval function shown in the image above will run the function myTimer() every 1second.The function myTimer() is shown below;



```
function myTimer()
 {
                timecounter--;
                document.getElementById("time").innerHTML=timecounter.toString();


                    if (timecounter===0)
                    {
                    gamestate=2;
                     var z = document.getElementById("loseraudio");
                     z.play();
                    window.alert("Time is up.You lose!! \n \n Final score is "+ score + " \n Number of tries:" + tries  );
                    clearInterval(myVar);
                    initializegame();

                    }



}
```

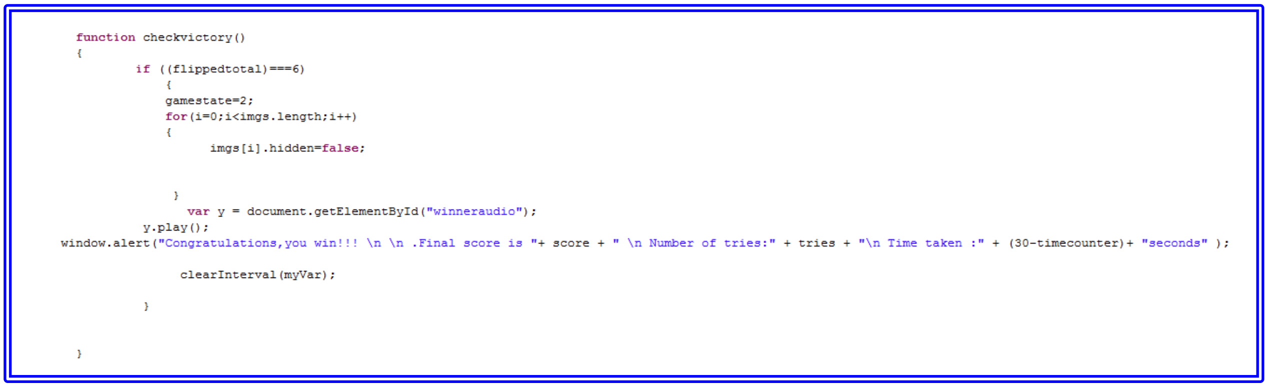
Fig9: myTimer() function

A global variable named timecounter was declared to store the value of the countdown timer. This variable is called timecounter. Every time this function is run, after every second that is, the timer is decremented by one. We then check if the game time has run out. If the time has run out, the user is informed that he/she has lost. The clearinterval() function stops the program executing 1second interrupts to update timer. This is essentially stopping the timer. The user interface is updated every time the timecounter value changes, so the user can see what time he/she has left in the game.

Requirement#7

When the user finishes the game, meaning all cards are face up, prompt a JavaScript alert to congratulate him/her. In your alert, show the score, time (duration), tries. Show everything in a single alert.

SOLUTION:



```
      function checkvictory()
      {
              if ((flippedtotal)===6)
                  {
                  gamestate=2;
                  for(i=0;i<imgs.length;i++)
                  {
                        imgs[i].hidden=false;


                   }
                     var y = document.getElementById("winneraudio");
               y.play();
    window.alert("Congratulations,you win!!! \n \n .Final score is "+ score + " \n Number of tries:" + tries + "\n Time taken :" + (30-timecounter)+ "seconds" );

                    clearInterval(myVar);

               }


      }
```

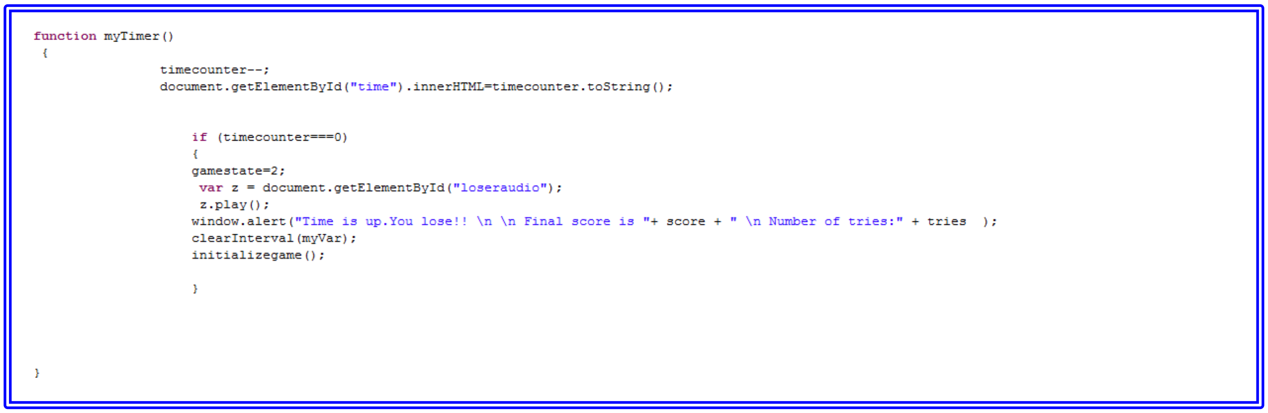
Fig10: victory message

The function checkvictory() is run every time a correct pair is flipped. If all pairs are flipped correctly, the function has a concatenated string which will display the score,tries and time taken in a single alert window.The window will also present a congratulatory message.

Requirement#8

If the timer reaches 0s before all pairs are found, prompt a JavaScript alert showing the score and tries

SOLUTION:



```
function myTimer()
 {
                timecounter--;
                document.getElementById("time").innerHTML=timecounter.toString();


                    if (timecounter===0)
                    {
                    gamestate=2;
                     var z = document.getElementById("loseraudio");
                     z.play();
                    window.alert("Time is up.You lose!! \n \n Final score is "+ score + " \n Number of tries:" + tries  );
                    clearInterval(myVar);
                    initializegame();

                    }



}
```

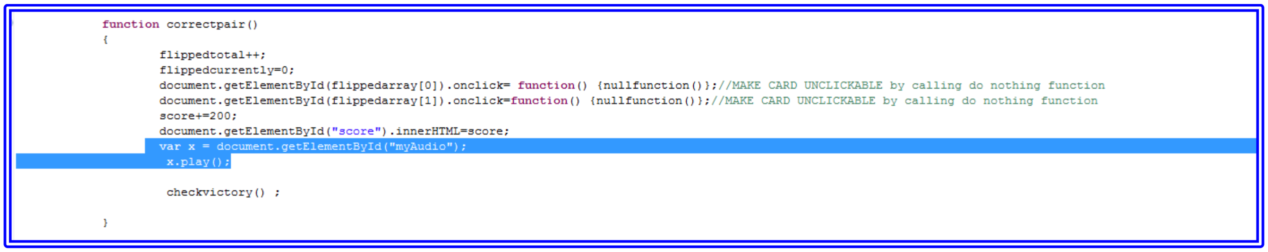
Fig11: game over message

Again inside the myTimer() function, we check if the timer has reached 0. If it has, an alert which contains a string concatenation of the score and tries is shown.

Requirement#9

Add a sound effect when the user finds a correct pair

SOLUTION:

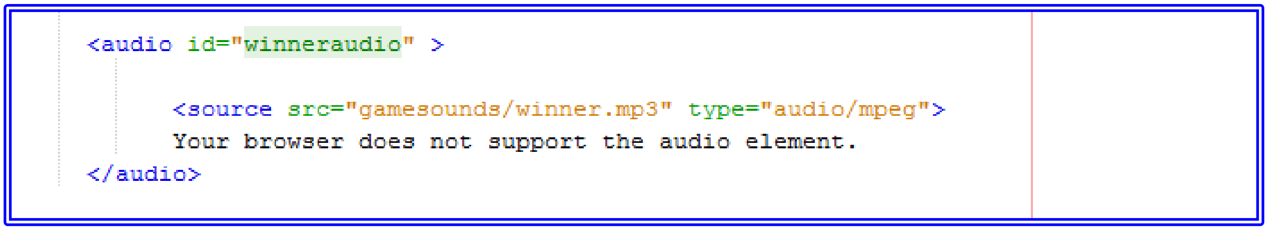


```
        function correctpair()
        {
                flippedtotal++;
                flippedcurrently=0;
                document.getElementById(flippedarray[0]).onclick= function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
                document.getElementById(flippedarray[1]).onclick=function() {nullfunction()};//MAKE CARD UNCLICKABLE by calling do nothing function
                score+=200;
                document.getElementById("score").innerHTML=score;
                var x = document.getElementById("myAudio");
                 x.play();

                 checkvictory() ;

        }
```

Fig12: playing winning sound



```
<audio id="winneraudio" >

      <source src="gamesounds/winner.mp3" type="audio/mpeg">
      Your browser does not support the audio element.
</audio>
```

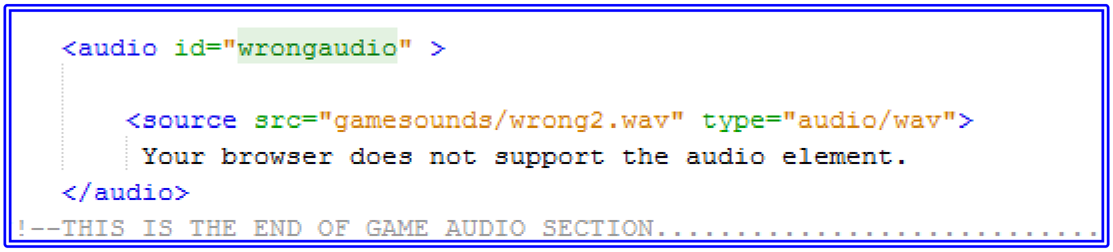
Fig13: winning sound audio object in HTML

An <audio> object was added to the original HTML file, and the .play() method was used to play the winning sound within the correctpair() function which only runs when there is a correct pair flipped up.

Requirement#10

Add a sound effect when the user selects an incorrect pair

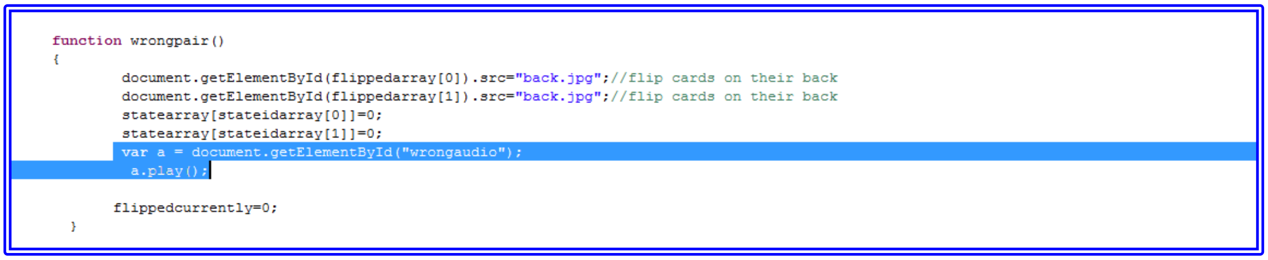
SOLUTION:



```
<audio id="wrongaudio" >

    <source src="gamesounds/wrong2.wav" type="audio/wav">
     Your browser does not support the audio element.
</audio>
!--THIS IS THE END OF GAME AUDIO SECTION............................
```

Fig14: wrongpair audio object in HTML



```
function wrongpair()
{
        document.getElementById(flippedarray[0]).src="back.jpg";//flip cards on their back
        document.getElementById(flippedarray[1]).src="back.jpg";//flip cards on their back
        statearray[stateidarray[0]]=0;
        statearray[stateidarray[1]]=0;
        var a = document.getElementById("wrongaudio");
         a.play();

       flippedcurrently=0;
  }
```

Fig15: playing losing sound

An <audio> object was created for the sound played when a player flips an incorrect pair. This object is acquired using the getElementById method and played using the play() method.

Design Summary

Program variables:

Global variable	function
imageArray	**Stores the location of images on the local hard drive.**
stateArray	**stores information about each card,**

	whether facing up or down
flippedArray	**Stores the IDs of the cards which are currently upwards flipped**
gamestate	**Stores the state of game whether ended, not started or in-game**
flippedcurrently	**Stores the number of cards that are currently upwards flipped**
flippedtotal	**Stores the number of pairs that have been correctly matched**
stateidarray	**Stores the statearray index of the 2 cards that are upward faced**
Shuffledarray	**Stores the cards that have been shuffled.**
Score	**Stores the player's current game score**
Tries	**Stores the number of card pairs the user has flipped both successfully and unsuccessfully**
Shiftindex	**Used as index variable when performing array shift operations**
Timecounter	**Stores the number of seconds the user has left of game time**
MyVar	**Return variable for setInterval function**

Imgs	**Array used to store collection of all image objectes on the webpage**

Program Functions

Function	**Purpose**
Initializegame	Set initial values of globals
InitializeUI	Show all cards face down
checkvictory	Check if user has won game
correctpair	Hide matched pair. Update globals.
wrongpair	Flip back wrong pair.Update globals
checkpair	Create delays for correctpaur and wrongpair functions and route to these functions appropriately
flipcard	Change the appearance of card based on statearray values, and update globals
Nullfunction()	Does nothing. Used when disabling flipped up cards
Shufflecards	Uses random number generator to shuffle cards
MyTimer	Keeps track of game time.

4. RESULTS

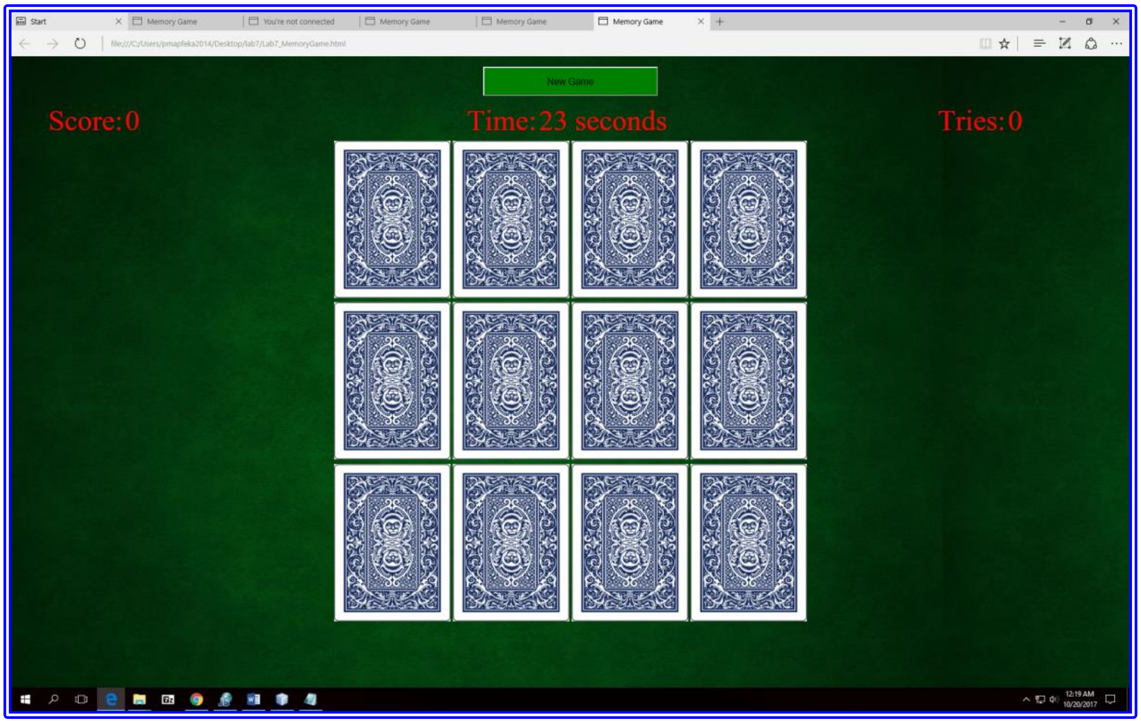
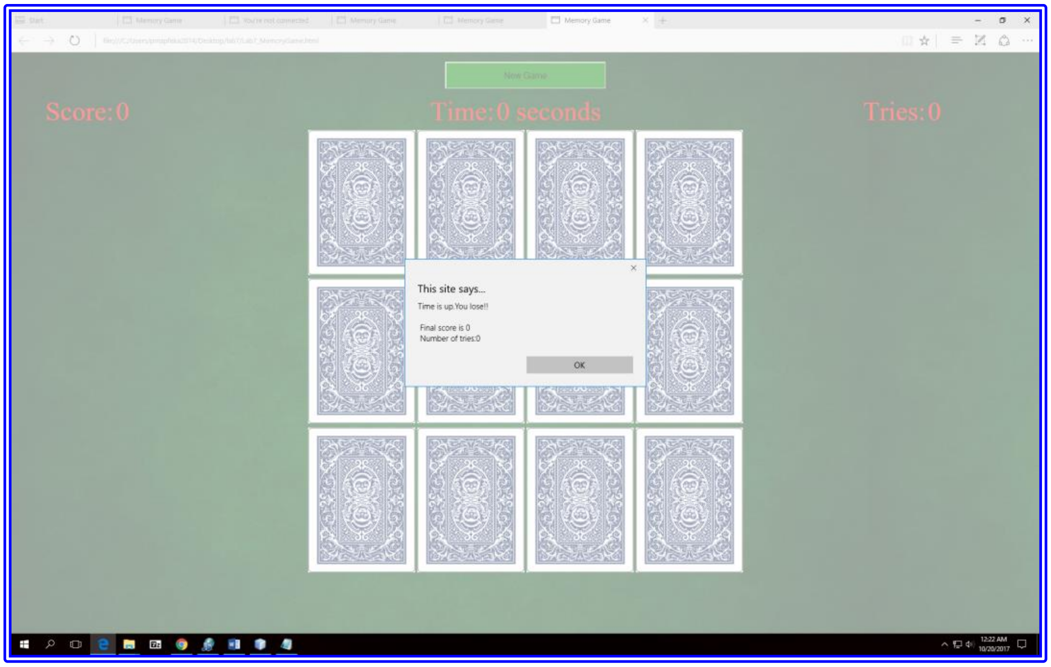


Fig16: final output1



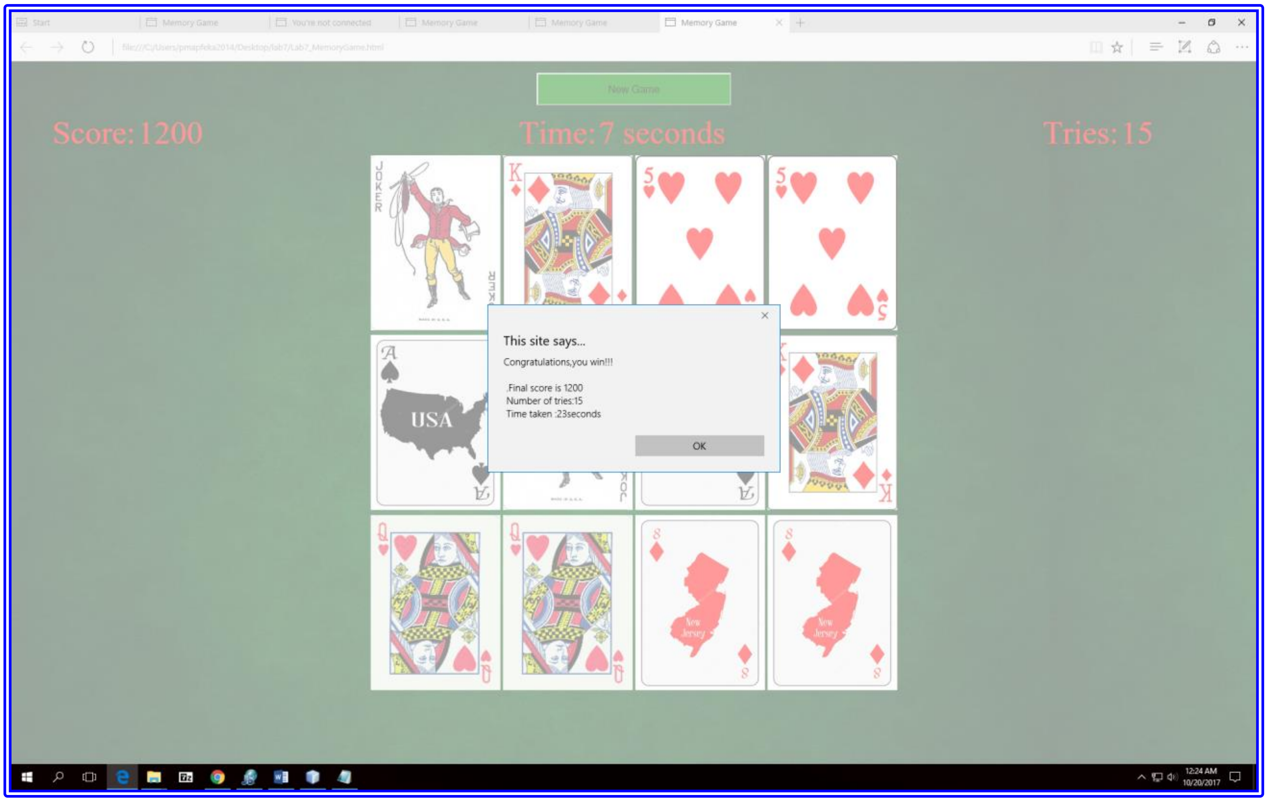
This site says...

Time is up.You lose!!

Final score is 0
Number of tries:0

OK

Fig17: final output2



Score: 1200

Time: 7 seconds

Tries: 15

This site says...

Congratulations,you win!!!

.Final score is 1200
Number of tries:15
Time taken :23seconds

OK

Fig18: final output3

5. <u>Conclusion</u>

The lab was a success. All objectives were met.