Final Project Part 2 - Memory Matching Game Code – Text Based Game

Requirements

- Create a **class** 'MemoryMatchGame', with the various variables, arrays and functions need to play the game.
- Hint: Use functions to divide the code up into specific functional areas

Have the game player select a 1 of the 3 Themes and have 50 words associated with the selected theme.

- Choose game theme:

  1) Theme name 1

  2) Theme name 2

  3) Theme name 3

  Creating the three(3) theme files is part 1 of the final project.

  - Words must have a common theme - your choice
  - MAX individual word length is 8 characters, no spaces, 1 word per line…
  - Examples: Like Periodic Table Elements, or Sports teams, or Types of cars…
  - Hint: load, from one of the three files, into a single dim array of string in class (Menu to select)

Have one Term describing a category you picked. This is the FACE term…

Additional Menu User Interaction:

- Level of Play – Use selects at <u>start</u> of game

  4 x 4 grid (Easy)

  6 x 6 grid (Moderate)

  8 X 8 grid (Difficult)

  Hint: Save as a variable in the class

- Speed of Play – At <u>start</u> of game, User selects time interval for User selected term-pair to display

  2 seconds (Difficult)

  4 seconds (Moderate)

  6 seconds (Easy)

Hint: Save as a variable in the class, for MS Window coders – use "sleep" code for delay…

Next, *Populate answer Grid with randomly selected Terms from the theme array*

- *At start of game – program places the same face/theme term in **ALL visible** squares in the visible grid*
  Real Answers not yet visible, only theme name is displayed in all squares, at start of game.
- *Program select number of random terms from the 50 available for selected theme (that programmer set up)*
  - *If 4 x 4 grid, randomly pick 8 terms, place each image name **twice** in 2-Dim array.*
  - *If 6 x 6 grid, randomly pick 18 terns, place each image name **twice** in 2-Dim array.*
  - *If 8 x 8 grid, randomly pick 32 terms, place each image name **twice** in 2-Dim array.*

  Hint: Randomly shuffle theme array and just pick the first 8, or 18 or 32 terms per game player selection

*Next, **display** the current game state on screen.*

Note: 'Answer' array is different from 'display' array

*During the course of play, the face/theme term in the display grid is **replaced** by a*

corresponding array terms, when user selects a grid square

Decide on how the user select/chooses a square/cell/location that is displayed… there many different methods.

Game Play

1) *User selects a FIRST square, the theme/face term in the grid square is replace with correspond stored term, from the 2-dim answer array*
2) *User selects a SECOND square, the term theme/face in the second grid square is replace with the corresponding stored term, from the 2-dim answer array*
3) *The computer compares the terms for the two selected squares.*
   *If they are the same, the terms remain on the screen and can **no** longer be selected.*
   If they are different, the term remain the screen for 2, 4 or 6 seconds, depending on user selection at the beginning of the game. After that elapse time, those two grid terms are replaced with the face/theme term.

==========

The class you write

A class consists of variables/arrays and functions.

All your variables/arrays and functions are to be encapsulated inside the Memory Match game

class you write.

The class will use 1 and 2 dimensional arrays

The class will have several variables

The class will have several functions – clearly named

There will be NO GLOBAL VARIABLES/Arrays or functions declared above int main(). All variables and arrays and functions will be ENCAPSULATED in the class.

The int main() in your code contain only two lines of code::

```
#include iostream;
using namespace std;
#include string;
#include MemoryMatchGame;
Int main() {
    MemoryMatchGame  Game1;  // first line - declare instance of game
    Game1.start();                    // second line - start game
}
```

Timer (Extra credit) - Create/display a timer that keep track of the number of seconds it took to win a game.

To receive the most credit, this project must be functional.

Hint:

Possible arrays needed:

- *string theme50Words[50] - load from file into this arrray*
- *string answerArray[#][#} - load 8, 18, or 32 words twice into this array, depending on game size selected*
- *string gamePlayArray[#}[#] – Game play and display array*