ECE 175 Homework Assignment 7

Due Date: by 11.59 pm on Tuesday October 26, 2021

Submission Instructions: Submit only .c flies (hw7p1.c, hw7p2.c) in the designated Assignment Dropbox on D2L.

Conventions: Name your C programs as *hw**x**p**y**.c* where ***x*** corresponds to the homework number and ***y*** corresponds to the problem number.

If each .c file is not properly named, 1 point will be deducted from your homework score.

Write comments in your programs. Programs with no comment will receive PARTIAL credits. Refer to previous homework handouts for minimum required comments in your program.

Topics: two-dimensional array and Strings

Problem 1(30 points): Write a C program that

- Lets user enter a statement,
- Reverses the words of the entered statement and display, and
- Displays number of words the entered statement contains.

Note: - fgets( ) should be used.
- assuming that each word is separated by a space.

Test case: birds and bees
are you as happy as I am
ECE 175 Class
fall leaves after leaves fall
The quick brown fox jumps over a lazy dog

Sample code execution 1: Bold entered by a user
Enter a statement: **birds and bees**
bees and birds
There are 3 words in the statement

Sample code execution 2: Bold entered by a user
Enter a statement: **The quick brown fox jumps over a lazy dog**
dog lazy a over jumps fox brown quick The
There are 9 words in the statement

Problem 2 (40 points) Interactive Memory (**sum to 9**) game

A memory (sum to 9) game is a matching game that starts with a set of digit cards facing down on the table. The player chooses two cards and turn them over. **If the digits from those cards are <u>summed to 9</u>, the two cards remain face up**. Otherwise, they are flipped face down. The game ends when all cards are faced up or a user decides to quit.

We will use two-dimensional (2D) array with 2 rows and 5 columns (2 x 5 2D array) to simulate a memory (sum to 9) game. The values from 0 – 9 represent the face of cards. See below for an example.

```
3 2 1 8 7
0 4 6 9 5
```

Write an C program to implement an interactive memory game such that

1) The game is first displayed with ? in each position.
2) It asks a user to enter a position of the first card that he/she wants to turn over. The row can only be from 1 to 2 and the column can only be from 1 -5. Your program will keep asking until a user enters valid values for row and column.
3) It then asks for a position of the second card. Your program will continue asking
   a) if values of row and column are not valid.
   b) If it is the same position as the first card in step 2)
4) If the positions that a user just entered correspond to the position(s) that the card(s) are already face up, repeat step 2) and 3) above
5) If the two valid positions are entered, the game is displayed with those 2 cards face up.
6) If the digits of those two cards are summed to 9, they remain face up. Otherwise, they are flipped face down.
7) It asks whether a user wants to continue. If yes, repeat step 2) – 6) above

The game ends when a user decides to quit or when all cards are faced up.

Your program should be modular and **must** use <u>at least</u> the following functions:

```
void print_board(int x[][5], int show[][5]);
```

if the element in the 2D array show is 1, print value of 2D array x at the same index (location). Otherwise, print ?

```
void generate_game(int x[][COL], int choice);
```

This function is to generate one memory (sum to 9) game. You should use it in your program. `The code for this function is given on the next page and you can use it in your program to generate a game.`

Note: If you call the generate_game function with choice = 1, the 2D array used for the memory game will be fixed (use this when developing your program).

```
//you can use the function below in your C program.
void generate_game(int x[][COL], int choice)
{
    int count = 0;
    int i, j, r, c;
    //fixed game - for code development (when choice = 1)
    //x = 3 2 1 8 7
    //    0 4 6 9 5
    if (choice == 1) {
        x[0][0] = 3; x[0][1] = 2; x[0][2] = 1; x[0][3] = 8; x[0][4] = 7;
        x[1][0] = 0; x[1][1] = 4; x[1][2] = 6; x[1][3] = 9; x[1][4] = 5;
    }
    else { // use rand()
        for (i = 0; i < ROW; i++)
            for (j = 0; j < COL; j++)
                x[i][j] = 100; //100 represent empty element

        for (i = 0; i < 10; i++) { //fill the board with value i from 0 -9
            while (1) {
                r = rand() % ROW;    //generate 0 - (ROW-1)
                c = rand() % COL;    //generate 0 - (COL-1)
                if (x[r][c] == 100) {
                    x[r][c] = i;
                    break;
                }
            }
        }
    }
}
```

After your code works and you want the memory game to be randomly generated, do as follows (Section 2.24 in zyBooks discusses random number generation):

a) In ***your .c file***, include the following header files (in addition to <stdio.h>)

    <stdlib.h> //this is for rand();
    <time.h> //this is for time();

b) In your ***main function***,

b.1) call the generate_game function with **choice = 2**.

b.2) srand statement

- when developing the program, write the following statement as the ***first statement*** in your main function (before declaration of variables)

`srand(1);`

/*1 in this function can actually be any integer. With a fix value of seed used with srand, it generates the same sequence of random numbers every time you run the program. It helps with debugging your program.

- After your program works correctly, change the statement to

`srand((int)time(0));` //this generate different value of seed //every time the program is run. It makes the game more random.

Sample code execution 1: **bold** information entered by a user

```
Let's play Memory (sum to 9) game
?  ?  ?  ?  ?
?  ?  ?  ?  ?
enter first position: row(1-2) and column(1-5): 4 8
enter first position: row(1-2) and column(1-5): 1 6
enter first position: row(1-2) and column(1-5): 3 4
enter first position: row(1-2) and column(1-5): 1 1
enter second position: row(1-2) and column(1-5): 1 1
first and second positions CANNOT be the same
enter first position: row(1-2) and column(1-5): 1 1
enter second position: row(1-2) and column(1-5): 3 6
enter second position: row(1-2) and column(1-5): 2 3
3  ?  ?  ?  ?
?  ?  6  ?  ?
3+6 = 9. Good job
continue (q to quit)? t
enter first position: row(1-2) and column(1-5): 1 1
3 already open at (row,col) = (1,1)
enter first position: row(1-2) and column(1-5): 1 2
enter second position: row(1-2) and column(1-5): 2 3
6 already open at (row,col) = (2,3)
enter first position: row(1-2) and column(1-5): 2 2
enter second position: row(1-2) and column(1-5): 2 5
3  ?  ?  ?  ?
?  4  6  ?  5
4+5 = 9. Good job
continue (q to quit)? a
enter first position: row(1-2) and column(1-5): 1 2
enter second position: row(1-2) and column(1-5): 2 1
3  2  ?  ?  ?
0  4  6  ?  5
2+0 = 2 (not 9)
3  ?  ?  ?  ?
?  4  6  ?  5
continue (q to quit)? q
The board is
3  2  1  8  7
0  4  6  9  5
Good Bye
```

Sample code execution 2: **bold** information entered by a user

```
Let's play Memory (sum to 9) game
? ? ? ? ?
? ? ? ? ?
enter first position: row(1-2) and column(1-5): 1 1
enter second position: row(1-2) and column(1-5): 4 7
enter second position: row(1-2) and column(1-5): 2 5
3 ? ? ? ?
? ? ? ? 6
3+6 = 9. Good job
continue (q to quit)? e
enter first position: row(1-2) and column(1-5): 1 2
enter second position: row(1-2) and column(1-5): 1 3
3 2 4 ? ?
? ? ? ? 6
2+4 = 6 (not 9)
3 ? ? ? ?
? ? ? ? 6
continue (q to quit)? w
enter first position: row(1-2) and column(1-5): 1 2
enter second position: row(1-2) and column(1-5): 2 4
3 2 ? ? ?
? ? ? 7 6
2+7 = 9. Good job
continue (q to quit)? y
enter first position: row(1-2) and column(1-5): 1 3
enter second position: row(1-2) and column(1-5): 2 3
3 2 4 ? ?
? ? 5 7 6
4+5 = 9. Good job
continue (q to quit)? o
enter first position: row(1-2) and column(1-5): 1 4
enter second position: row(1-2) and column(1-5): 1 5
3 2 4 8 1
? ? 5 7 6
8+1 = 9. Good job
continue (q to quit)? r
enter first position: row(1-2) and column(1-5): 2 1
enter second position: row(1-2) and column(1-5): 2 2
3 2 4 8 1
9 0 5 7 6
9+0 = 9. Good job
Congratulations, You win the game
Good Bye
```