Lab Activity #7 – 2D Arrays and File Streams

Exercise #1:

Write a program that declares a two-dimensional array named ***myFancyArray*** of the type ***double.*** Initialize the array to the following values:

23	14.12	17	85.99
6.06	13	1100	0
36.36	90.09	3.145	5.4

1. Create a function that will return the sum of all elements in the array. Call this function from main and print out the result.
2. Create a function that will use a nested loop to display all the elements in the array to the screen. Call this function from main.

Exercise #2:

Write a program that can be used to assign seats for a commercial airline. The airplane has 13 rows, with 6 seats in each row.

Create a menu-driven program so that the user (in a loop) can choose to display a map of the seats, or choose a seat.

Menu option 1: Display a "map" of all of the seats on the airplane (display a star (*) to indicate the seat is available; display an X if the seat is occupied).

Menu option 2: Ask the user and to select the specific seat he wishes to reserve. Either reserve the seat, or tell the user that the request could not be completed.

You must create this program by writing the following functions:

displayMap will display the current seating map of the entire airplane
makeReservation will let the user select the specific seat(s) they wish to reserve, call a function validateFunction, and reserve the seat it is available. **validateFunction** will take the user's selection as a parameter and will return true if the user can reserve the seat and will return false if the seat is already taken.

	A	B	C	D	E	F
Row 1	*	*	X	*	X	X
Row 2	*	X	*	X	*	X
Row 3	X	*	X	X	*	X
(etc.)						

Exercise #3:

(NOTE: You may read the file only once in your program!)

You are the owner of a hardware store and need to keep an inventory that can tell you what different tools you have, how many of each you have on hand and the cost of each one. The data is organized as follows:

- The first column contains the tool name
- The second column contains the quantity of that product in inventory.
- The third column contains the cost per item or unit price of that product.

Use the following information to start your file:

Hammer	76	11.99
Jigsaw	21	14.99
Wrench	10	4.49

Write a C++ program that will do the following:

- Read data from a data file you create called ***hardware.dat*** located on the hard drive. The inventory changes occasionally, so you do not know what items are in the file or how much data there is to process.

- Call a function that will return the total value of the store's inventory for any one particular product (unit price * number of items).

- Compute the total number of items in the inventory.

- Print the output to a file you create called ***hardware.out*** to display the results. The output in that file should be displayed as follows (make sure you include the necessary headings that are shown below):

MY HARDWARE STORE

TOOL	QUANTITY	PRICE
Hammer	76	11.99
Jigsaw	21	14.99
Wrench	10	4.49

Value of all hammers in stock:	$ 911.24
Value of all jigsaws in stock:	$ 314.79
Value of all wrenches in stock:	$ 44.90

Total items in inventory: 107

Exercise #4: Tic-Tac-Toe (courtesy of Dr. Huo)

The following program is the base for the game: tic-tac-toe. The main function and program structure is given, don't modify them. Complete three functions as required. Compile and run this program, and then you can play the game. Have fun!

```
#include<iostream>
using namespace std;

const int DIM=3;
char chessboard[DIM][DIM];
//initChessBoard
void initChessBoard(char cb[][DIM])
{
        //set all the elements of the ChessBoard to blanks
        //Complete this part in the following:

}

//printChessBoard
void printChessBoard(char cb[][DIM])
{
        //print all the elements of the chessBoard with each row in one line
        //Complete this part in the following:

}

//putChequer
bool putChequer(char cb[][DIM], int i, int j, char c)
{
        /* if i and j are not out of bound(that is, i and j are in the range of 0 and DIM-1) and
cb[i][j] is not occupied(that is, the value of cb[i][j] is blank), set cb[i][j] to be the value of c and
return true. Otherwise, return false.*/
        // Complete this part in the following:

}
//judge the state of the game. The player has put c (which is either X or O) in the position (row,
col).
// If all the elements in this row are c, c wins
// If all the elements in this column are c, c wins.
// If c is in the main diagonal, and if all the elements in the main diagonal are c, c wins.
// If c is in the opposite diagonal, and if all the elements in the opposite diagonal are c, c wins.

bool state(char cb[][DIM], int row, int col, char c)
{
```

```
        /* We declare four variables count1, count2, count3, count4 to represent the
        occurrence number of c in row number row, column number col, in the main diagonal,
        and in the opposite diagonal. If after calculation, count1, count2, count3 or count4
        equals to DIM, return true(that is, c wins). */

        int count1=0, count2=0, count3=0, count4=0;

        for(int i=0; i<DIM; ++i)
        {
                // Complete this part in the following:
                // if the element in position (row, i) is c, count1 is increased by 1.

                // if the element in position (i, col) is c, count2 is increased by 1.

                /* if c is in the main diagonal, and the element in position (i, i) is c,
                count3 is increased by 1.*/

                /* if c is in the opposite diagonal, and the element in position (i, DIM-1-I) is c,
                count4 is increased by 1.*/
        }
        return (count1==DIM || count2==DIM || count3==DIM || count4==DIM);
}


int main()
{
        int row, col;
        int blanks=DIM*DIM;
        initChessBoard(chessboard);
        printChessBoard(chessboard);
        char cur='O';

        cout<<"Input the position(row col), (-1 -1) for exit; It is the turn of "<<cur<<endl;
        cin>>row>>col;

        while(row!=-1 && col!=-1)
        {
                if(!putChequer(chessboard, row, col, cur))
                {
                        cout<<"Invalid move"<<endl;
                        printChessBoard(chessboard);
                }
                else
                {
                        --blanks;
                        printChessBoard(chessboard);
                        if(state(chessboard, row, col, cur))
```

```
                        {
                                cout<< cur << " Wins"<<endl;
                                return 0;
                        };
                        if(blanks==0)
                        {
                                cout<< "Ties"<<endl;
                                return 0;
                        }
                        if(cur=='X')
                                cur= 'O';
                        else cur='X';
                }
                cout<<"Input the position(row col), (-1 -1) for exit; It is the ture of "<<cur<<endl;
                cin>>row>>col;
        }
}
```

Exercise #5: Complete Your Game

Add tic-tac-toe to your game!!