# Homework Lab 7: File Streams, Arrays

***Exercise #1:***

Write a short program that reads in students’ test scores FROM A FILE ON DISK. (these can be any number between 0-100.) Store the grade values in an array.

Use the following data: 45 67 89 90 89 56 67 68 69 99 97 76 77 79 89

* Compute and print the average grade.
* Add 10 points to any student whose grade is below the average grade.
* Add 5 points to any student whose grade is above the average grade.
* Add 7 points to any student whose grade is exactly the average grade.
* Using the new updated grades: print out the number (count) of students who received over 90.

***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. The user can choose one of the following:

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).
2. Reserve a seat: Ask the user to select the specific seat(s) they wish 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** that will allow the user select the specific seat(s) they wish to reserve, and reserve it or tell the user that it already is in use.

The display looks like this:

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: Tic-Tac-Toe (courtesy of Dr. Huo) When you are done, add this to your game program!!***

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 x is in the main diagonal, and the element in position (i, i) is c,

count3 is increased by 1.\*/

/\* if x 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;

}

}