CS 210 – Fundamentals of Programming I
Spring 2007 – In-class Exercise for 3/12/07 & 3/13/07

Name(s):

(15 points) Complete this exercise in pairs. Write the answers to the written part in this assignment sheet and turn it in with a print out the program when you are done. Today's in-class exercise is to work with two-dimensional arrays. Here is a problem statement for today's program, and the analysis and design for the main program and the function PlaceMove.

Problem Statement
Write a tic-tac-toe game server that allows two users to play and keeps track of the moves made by each. It should not allow invalid moves, and it should display the board after every turn. It should determine when someone wins.

Behavior
In the game of tic-tac-toe, two players, named X and O, take turns marking positions on a 3 by 3 grid. A player wins a game of tic-tac-toe if he or she is able to mark three adjacent positions in any orientation (i.e., a whole row, a whole column, or either diagonal). If all positions are marked without a producing a winner, the game is a draw.

The program should initialize the board so that the board positions are numbered (as characters) like so:

1 2 3
4 5 6
7 8 9

It then should ask Player X and Player O to take turns choosing positions. Player X always goes first. The program does not allow players to choose positions that are out of range or that have already been marked. When a position is chosen, the position number should replaced by the player's character, and the changed board displayed. For example, the board in the middle of a game might look like:

X X O
4 5 6
O 8 9

The program should check for a winner after each turn and print out the an appropriate message when there is a winner or if the game ends in draw.
Analysis & Design of Main Program

Analysis - what data is constant, input, computed, or output?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>board size (3)</td>
<td>int</td>
<td>constant</td>
<td>BOARD_SIZE</td>
</tr>
<tr>
<td>maximum positions (9)</td>
<td>int</td>
<td>constant</td>
<td>MAX_POSITIONS</td>
</tr>
<tr>
<td>maximum turns (9)</td>
<td>int</td>
<td>constant</td>
<td>MAX_TURNS</td>
</tr>
<tr>
<td>tic-tac-toe board</td>
<td>char [ ][ ]</td>
<td>variable</td>
<td>theBoard</td>
</tr>
<tr>
<td>current player</td>
<td>char</td>
<td>variable</td>
<td>player</td>
</tr>
<tr>
<td>winning player</td>
<td>char</td>
<td>variable</td>
<td>winner</td>
</tr>
<tr>
<td>position chosen</td>
<td>int</td>
<td>variable</td>
<td>position</td>
</tr>
<tr>
<td>number of turns taken</td>
<td>int</td>
<td>variable</td>
<td>turns</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Greet user using Instructions()
2. Initialize game board using InitializeBoard(theBoard)
3. Initialize player to 'X', winner to '­', turns to 0
4. Display game board using PrintBoard(theBoard)
5. While winner is '­' and turns < MAX_TURNS do
   5.1. Read in position
   5.2. If PlaceMove (theBoard, player, position) is true then // valid move
      5.2.1. Compute winner = CheckForWinner(theBoard)
      5.2.2. Increment turns
      5.2.3. If player is 'X' then (switch players)
         5.2.3.1. Set player to 'O'
      5.2.4. Else
         5.2.4.1. Set player to 'X'
   5.3. Else // invalid move
      5.3.1. Print error message
   5.4. Display game board using PrintBoard (theBoard)
6. If winner is not '­' then
   6.1. Display winner
7. Else
   7.1. Display message that game was a draw

03/05/07
Analysis & Design of Function PlaceMove

Analysis - what data is received, passed back, returned, or local?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>game board</td>
<td>char [ ][ ]</td>
<td>variable</td>
<td>received &amp; passed back</td>
<td>board</td>
</tr>
<tr>
<td>current player</td>
<td>char</td>
<td>variable</td>
<td>received</td>
<td>player</td>
</tr>
<tr>
<td>position chosen</td>
<td>int</td>
<td>variable</td>
<td>received</td>
<td>position</td>
</tr>
<tr>
<td>validity of move</td>
<td>bool</td>
<td>variable</td>
<td>returned</td>
<td>valid</td>
</tr>
<tr>
<td>row index of position</td>
<td>int</td>
<td>variable</td>
<td>local</td>
<td>row</td>
</tr>
<tr>
<td>column index of position</td>
<td>int</td>
<td>variable</td>
<td>local</td>
<td>column</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Initialize valid to true
2. If position < 1 or position > MAX_POSITIONS then // out of range
   2.1. Set valid to false
3. Else
   3.1. Compute row = (position - 1) / 3
   3.2. Compute column = (position -1) %3
   3.3. If board[row][column] is 'X' or 'O' then // it's already taken
       3.3.1. Set valid to false
   3.4. Else // place move
       3.4.1. Set board[row][column] = player
4. Return valid

Assignment

0. Create a new project, the download file inclass16.cpp from the course webpage under today's date and add it to your project. Build and run this program. The call to function CheckForWinner is missing. The main program currently sets the winner variable to character '-' indicating there is no winner, so the program never declares a winner. However, you can have each player choose positions and see how the board fills up. You can also test the error handling. When the maximum number of turns have been taken without a winner, the program will declare a draw.
1. (5 points) Answer the following questions about this program.

   a. Write the variable declaration in the main program that is for a multi-dimensional array.

   b. What is the element type of the multi-dimensional array?

   c. How many (total) elements are there in the multi-dimensional array?

   d. What are the possible index values of the rows of multi-dimensional array? Of the columns?

   e. Write a C++ statement in the main program that would print out the element in tic-tac-toe board position 7.

2. (10 points) The analysis for the function CheckForWinner is:

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>game board to be evaluated</td>
<td>char [ ][ ]</td>
<td>variable</td>
<td>received</td>
<td>board</td>
</tr>
<tr>
<td>winner of game</td>
<td>char</td>
<td>variable</td>
<td>returned</td>
<td>winner</td>
</tr>
</tbody>
</table>

   The function should return the character `X` if Player X is the winner, the character `O` if Player O is the winner, or `-` if there is no winner.

   a. Write the prototype and definition for function CheckForWinner. Some observations that may help you:

      - You only have to check that any two pairs of the three positions in a possible winning combination are the same character, since the transitive property of equality implies the third pair will be the same character.
      - You can check for the equalities first, and if both are found to be true, then you can
determine the winning player by looking at the value of any of the elements in the winning combination.

- There are 8 possible winning combinations: each of 3 rows, each of 3 columns, and 2 diagonals.
- Make sure the function returns a computed result in all cases, including when there is no winner.

b. Change the main program to call CheckForWinner as indicated in the comments.

Rebuild and test your program until you are confident that it computes the correct result.

When you have completed this exercise, print out your program file and turn it in with one copy of this exercise sheet with your answers to the questions.