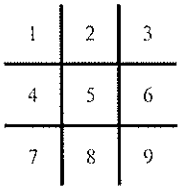
11/9

CS 504

ASSIGNMENT #5

Tic–tac–toe is a game played on a 3×3 grid. We number the grid elements as shown, to facilitate communication between the computer program and the players:



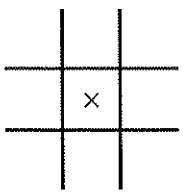
Here is how you might display the grid using asterisks and spaces:

```
        *         *
        *         *
        *         *
* * * * * * * * * * * * * * * * * * *
        *         *
        *         *
        *         *
* * * * * * * * * * * * * * * * * * *
        *         *
        *         *
        *         *
```

1. Write a Java program that supports a game of tic–tac–toe played by two people.

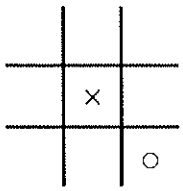
   The program should behave as follows:

   A. Ask Player 1 for the element number that he/she wants to mark. Then display a grid showing that element marked with an ×. For example, if Player 1 selects element 5, the program displays:



   B. Next, ask Player 2 for the element number that he/she wants to mark. Then, adding an additional mark – an o in the element Player 2 has just selected – display an updated grid. For example, if Player2 selects element 9 following Player 1's having selected

element 5, the program displays:



C. Repeat steps A and B, each time adding a new mark to the grid, until a player enters "$", signaling that the game is finished.

2. Modify your program so that it enforces these rules of tic–tac–toe:

   A. Only unmarked elements can be chosen by players.

   B. The game concludes automatically, without a player entering "$", when all nine elements are marked. (The game ends in a draw, called a "cat's game.")

3. Further modify your program to automatically detect that a player has won. (Display who won.) There is no need now for a player to enter "$" to signal that the game is finished.

4. Further modify your program so that the computer is Player 2. (Use any strategy you want when choosing an element; you don't have to try to win.)

5. Further modify your program so that the computer can be either player. (Begin by asking the user if he/she wants to be Player 1 or Player2; the computer is then the other player.)

6. (Extra credit) Further modify your program so that the computer tries to win.

7. (Extra Extra credit) Further modify your program so that the computer plays itself.

Structure your program so that it includes the methods shown here, many or all of which are called by method *main* to conduct the game. (Method *main* should be little more than calls to these methods.)

void *initializeGame* ( ) /* This method may be unnecessary in your implementation. */

int *getMove* (**int** *player*) /* Returns the position (1–9) of the element chosen by *player* in the next move. */

char *getTurnMark* ( ) /* Returns marking of next turn. (This might be implemented by calling *getTurn*.) */
/* (You may choose the return value to be of a type other than **char**.) */

int *getTurn* ( ) /* Returns 1 if it's Player's 1 turn, and 2 if it's Player's 2 turn. */

boolean *elementMarked* (**int** *element*) /* 0 < *element* < 10 */
/* (This might be implemented by calling *getMark*.) */

char *getMark* (**int** *element*) /* (You may choose the return value to be of a type other than **char**.) */

void *setMark* (**int** *element*, **char** *mark*)
/* (You may choose *mark* to be of a type other than **char**.) */

int *unmarkedElement* ( ) /* Returns the position (1–9) of an as-yet unmarked element. */

boolean *won* (**char** *mark*) /* (You may choose *mark* to be of a type other than **char**.) */

boolean *catsGame* ( )

void *displayBoard* ( )