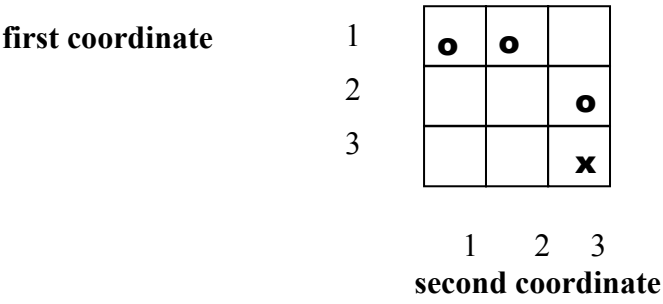
This problem concerns the writing two games, one called **Exam1** and one called **Exam2** to be played by two players. The games are played on a three by three board. The aim of each game is to end up with the most pieces on the board when the board is full. At the beginning of the game the board has an **x** in (1,1) and an **o** in (3,3). An empty square is represented by the character `'.'`. For the **Exam1** game each player places one piece in turn on any free space on the board. The game **Exam2** is an extension to the **Exam1** game. In **Exam2** a player can only place a piece if the square is adjacent (vertically or horizontally) to a square already occupied by the player. So on the board below an **x** can only be placed in (3,2), whereas if it is **o**s turn the only spaces that an **o** cannot be placed in are (3,1) and (3,2). All pieces that are held by an opponent in both the row and the column where the new piece is placed are then converted into the current player's piece. If **o** puts their piece in (1,3) then after the move **o** will have two new pieces (1,3) and (3,1). It doesn't matter if there is free space between the current players piece and the opponents piece - it is still captured.




You are given the following class, methods and strings to use in your program so that it can be autotested. **Do not print out any strings other than the ones below**:

```
class Coord{
   int row;
   int col;
}
void main()
void printBoard(char[][] board)
String prompt = "Please type in your move, row followed by column -> "
String tieMessage = "Its a tie!";
String firstWinner = "The first player is the winner.";
String secondWinner = "The second player is the winner.";
String occupied = "Space occupied.";
String offBoard = "Numbers not on board.";
String forfeit = "You forfeited your move.";
```

Answer the questions using Kenya or Java with assertions where appropriate. There will be a 5% mark penalty for non-compilation. Make sure you include assertions where appropriate. You may use the code provided and any of your code that you have written for earlier parts of the question. Before you start coding, you may find it helpful to examine the `main` method given.

1. Write a predicate (`boolean` method) `continuePlaying` that takes a board as a parameter and returns `true` if and only if there is at least one free space on the board. A free space is represented by the character `'.'`.

2. Write a method `getMove` that takes a board as a parameter, prompts (using `prompt`) for coordinates, reads in the user's input and returns the coordinates that will access the board. If the square is occupied then the user should be prompted again until a non-empty square's coordinates are entered. (You can assume that the coordinates typed in are on the board.)

3. Write a `void` method `placePiece` that takes a player (`'X'` or `'O'`), a pair of coordinates and a board as parameters and alters the board by putting the player's piece on the board at the coordinates given.

4. Write a method `switchPlayer` that takes as a parameter a character that is either `'X'` or `'O'` and returns the other one.

5. Write a `void` method `winningMessage` that takes a board and prints on the screen the appropriate message from `tieMessage`, `firstWinner` and `secondWinner`.

6. Now using the methods you have written and the given methods `printBoard` and `main` write a program that plays the **Exam1** game. Put your program into `Exam1.k` or `Exam1.java`.

7. Write a program that enables two players to play the the game **Exam2**. There are many ways to implement the game. Here are some hints which you might wish to use:

    a. Extend `getMove` to explicitly reject coordinates not on the board (before reprompting for input).

    b. You may decide to replace `placePiece` with code that consists of several methods. These could also

        i. check that there is a piece of the player next to the square where you wish to place, remembering that if there isn't a valid move then the turn is forfeited,

        ii. convert the pieces in the row and column to the player's piece.

    Put your completed program into a file called `Exam2.k` or `Exam2.java`. You may use code from the Exam1 game if you wish.

*Parts 1-6 carry 10%, of the marks each. Part 7 carries 40% of the marks.*