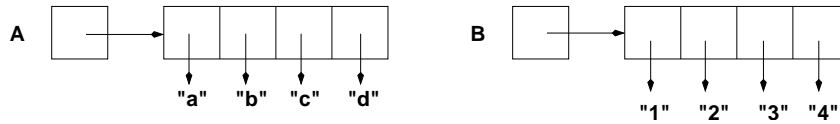


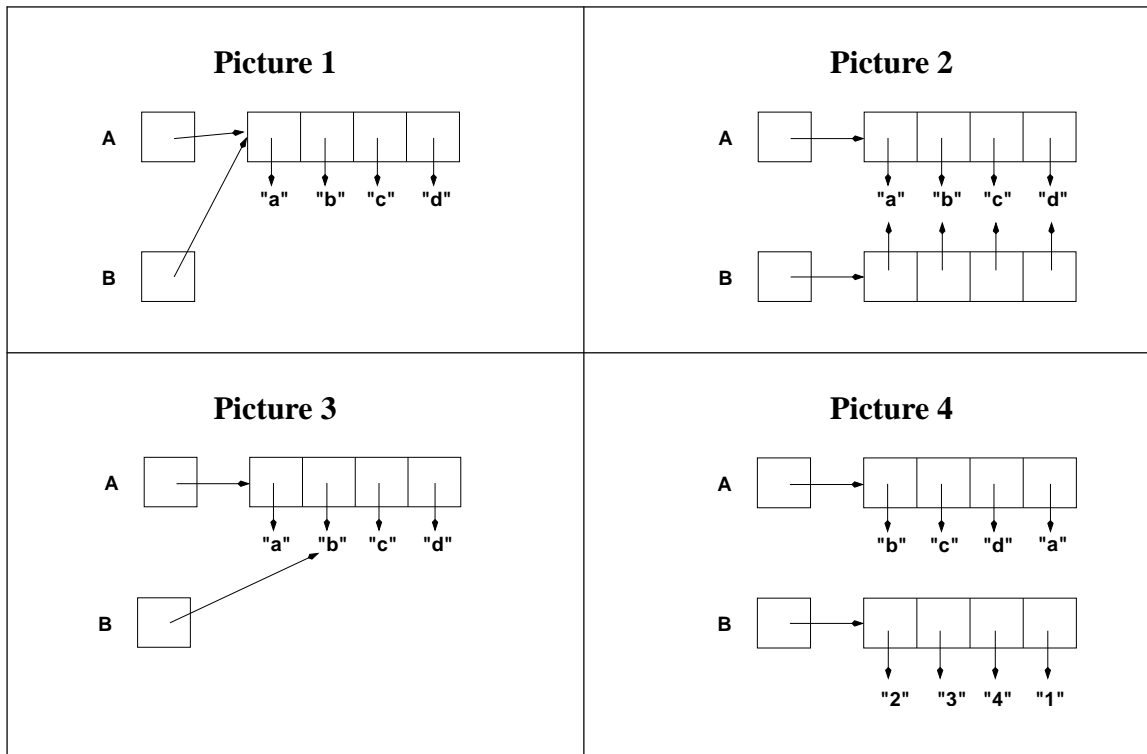
# WES-CS GROUP MEETING #9

## Exercise 1: Arrays and Aliasing

Assume that you have two *String* arrays, A and B, initialized as shown below.



For each of the pictures shown below, write code that would change arrays A and B from their initial values as shown above, to the new value shown in the picture. If there is no such code, explain why.



## Exercise 2: Designing and Implementing a Class

Below is an incomplete *CheckerBoard* class. When the main method executes, it should print the following output (representing a checkerboard with red and black squares):

```
R B R B R B R
B R B R B R B
R B R B R B R
B R B R B R B
R B R B R B R
B R B R B R B
R B R B R B R
```

Notice that the code uses a *CheckerBoardSquare* class. Your job is to write the *CheckerBoardSquare* class as well as the rest of the *CheckerBoard* class. To do this, you'll need to use the code given below to figure out what fields and methods each of the two classes needs to have, and what each method needs to do.

```
class CheckerBoard {

    // YOU MUST ADD CODE HERE

    public static void main(String[] args) {
        for (int i = 0; i < HEIGHT_OF_BOARD; i++) {
            for (int j = 0; j < WIDTH_OF_BOARD; j++) {
                CheckerBoardSquare square = new CheckerBoardSquare(i, j);
                square.printColor();
                System.out.print(' ');
            }
            endRow();
        }
    }
}
```

### Exercise 3: Two-Dimensional Arrays

Assume that the following *Person* class has been defined.

```
public class Person {
    private String name;
    private int birthYear;
    private String eyeColor;

    // constructor
    public Person(String nme, int brthYr, String eye) {
        name = nme;
        birthYear = brthYr;
        eyeColor = eye;
    }

    // accessor methods
    public String getName() { return name; }
    public int getBirthYear() { return birthYear; }
    public String getEyeColor() { return eyeColor; }
}
```

First, choose four people in your group and draw an array called *people* that contains four *Person* objects representing those four people.

Now (in groups of two or three) play a game of *concentration* using the green and orange cards. Each green card has a Java expression, and each orange card has a value. Start with all of the cards upside down. When it's your turn, you turn over one green card and one orange card. If they match you keep those two cards, and keep going; otherwise, your turn is over.

The game ends when you run out of green cards or the remaining orange cards don't match any green cards.

Whoever has the most cards wins!

## Exercise 4: Logical Thinking

Today's logical-thinking exercise is an old chess puzzle. The board (shown below) is a 3x3 part of the chessboard. The goal is to swap the red and blue knights, using a sequence of legal moves (of course, two knights may not occupy the same square at any time).

**B**: blue knight

**R**: red knight

|          |  |          |
|----------|--|----------|
| <b>B</b> |  | <b>B</b> |
|          |  |          |
| <b>R</b> |  | <b>R</b> |

If you can solve the puzzle, consider the following additional questions:

1. What is the minimum number of moves required to solve the puzzle?
2. Is it possible starting with the original configuration to arrange the knights as shown below?

|          |  |          |
|----------|--|----------|
| <b>R</b> |  | <b>B</b> |
|          |  |          |
| <b>B</b> |  | <b>R</b> |

```
people[1].getBirthYear()
```

```
(people[0].getEyeColor().compareTo(people[3].getEyeColor())) < 0
```

```
people[people.length-1].getBirthYear() > 1980
```

```
people[2].getName().size() < 6;
```

```
(people[2].getName().substring(0,1)).equals((people[3].getEyeColor().substring(0,1)))
```

```
people[1].getEyeColor().charAt(1)
```

1987

1986

1985

1984

true

false

true

false

true

false

true

false

l

r

a