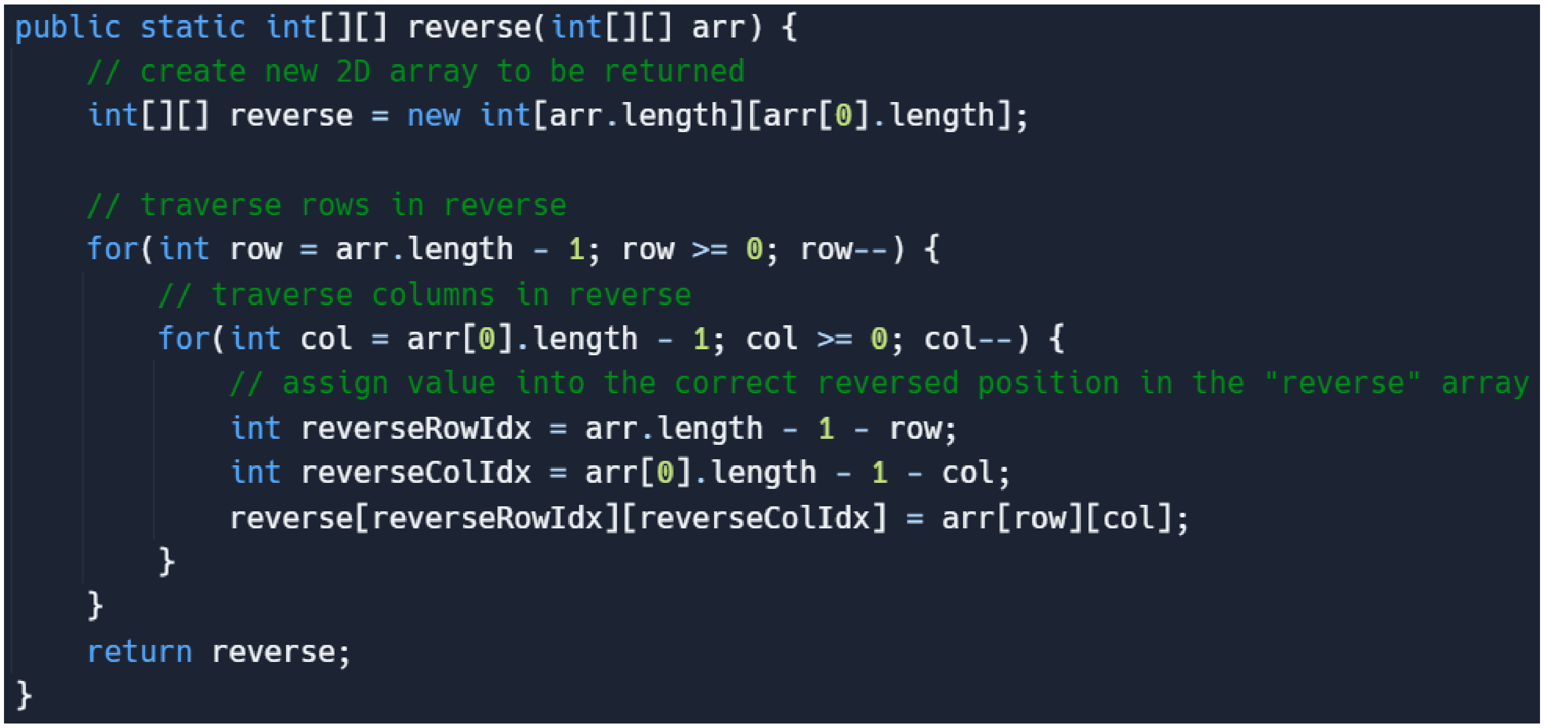
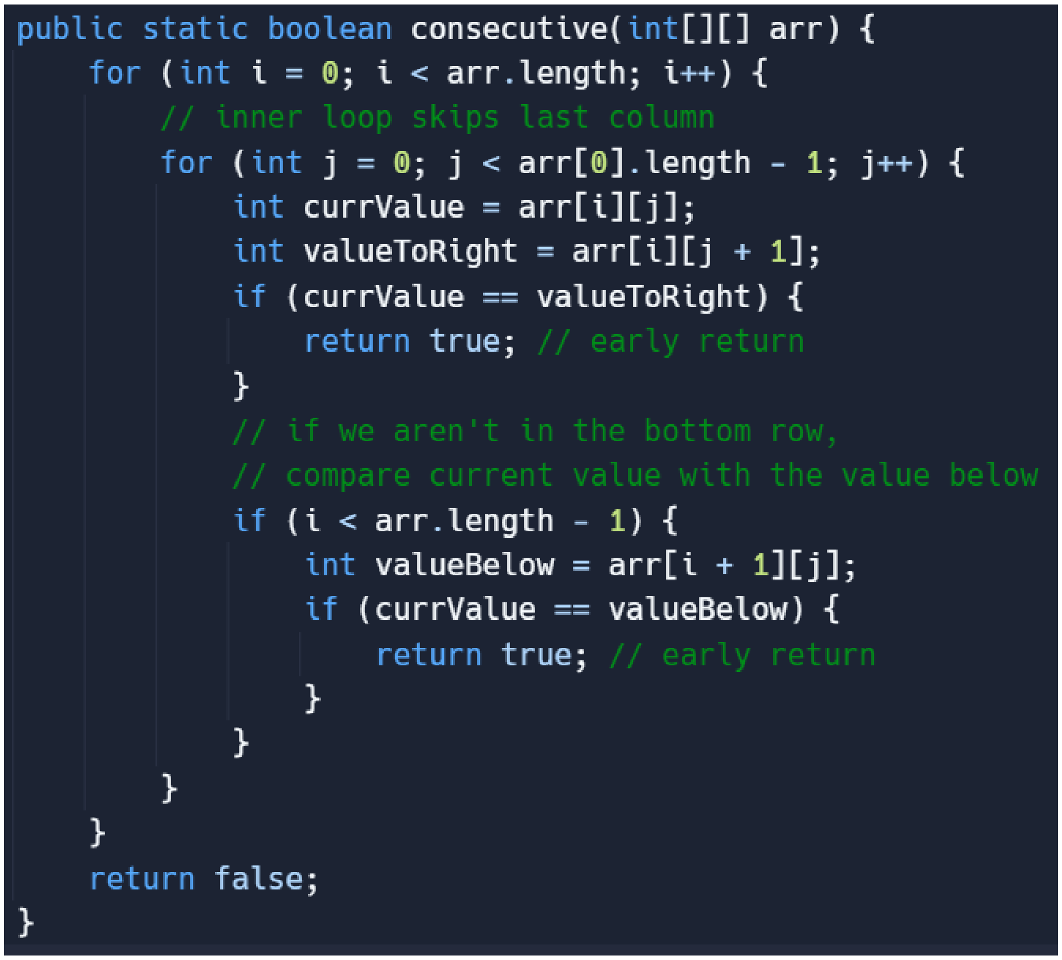
Sample Solutions for the Challenge Problems (last 3 methods of the lab sheet)

`reverse`

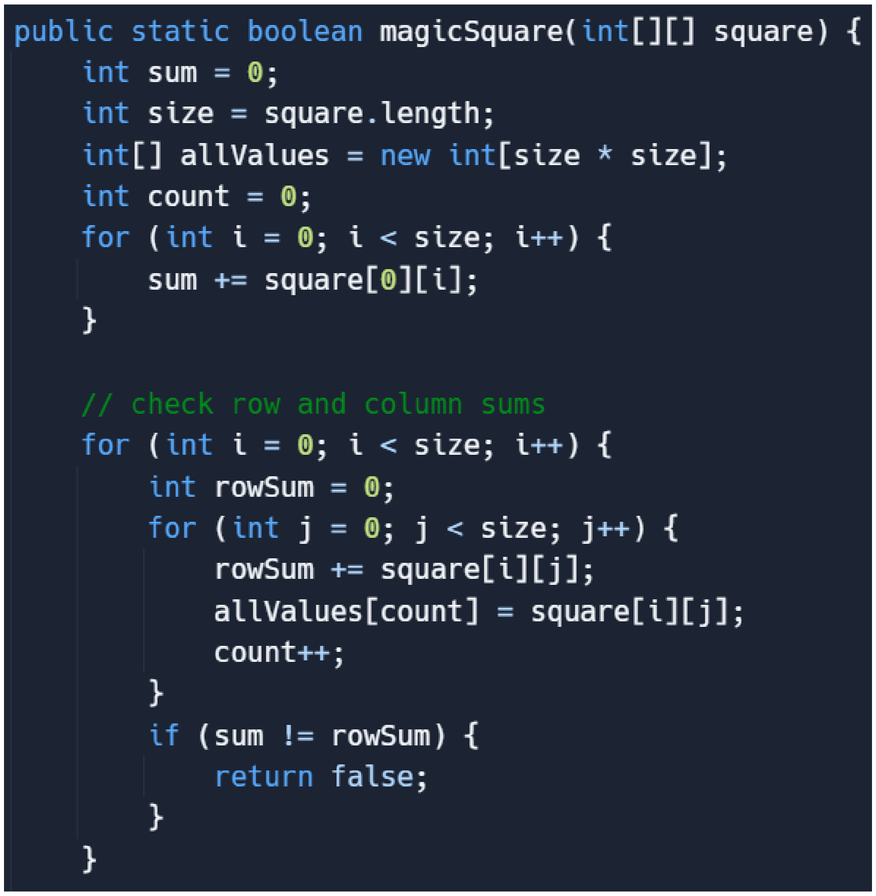


```java
public static int[][] reverse(int[][] arr) {
    // create new 2D array to be returned
    int[][] reverse = new int[arr.length][arr[0].length];

    // traverse rows in reverse
    for(int row = arr.length - 1; row >= 0; row--) {
        // traverse columns in reverse
        for(int col = arr[0].length - 1; col >= 0; col--) {
            // assign value into the correct reversed position in the "reverse" array
            int reverseRowIdx = arr.length - 1 - row;
            int reverseColIdx = arr[0].length - 1 - col;
            reverse[reverseRowIdx][reverseColIdx] = arr[row][col];
        }
    }
    return reverse;
}
```

`consecutive`

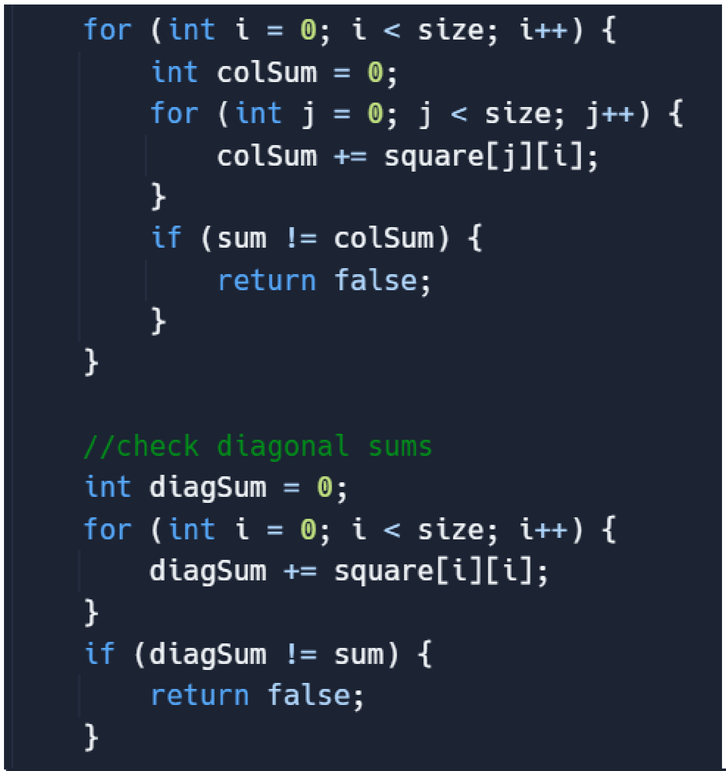


```java
public static boolean consecutive(int[][] arr) {
    for (int i = 0; i < arr.length; i++) {
        // inner loop skips last column
        for (int j = 0; j < arr[0].length - 1; j++) {
            int currValue = arr[i][j];
            int valueToRight = arr[i][j + 1];
            if (currValue == valueToRight) {
                return true; // early return
            }
            // if we aren't in the bottom row,
            // compare current value with the value below
            if (i < arr.length - 1) {
                int valueBelow = arr[i + 1][j];
                if (currValue == valueBelow) {
                    return true; // early return
                }
            }
        }
    }
    return false;
}
```

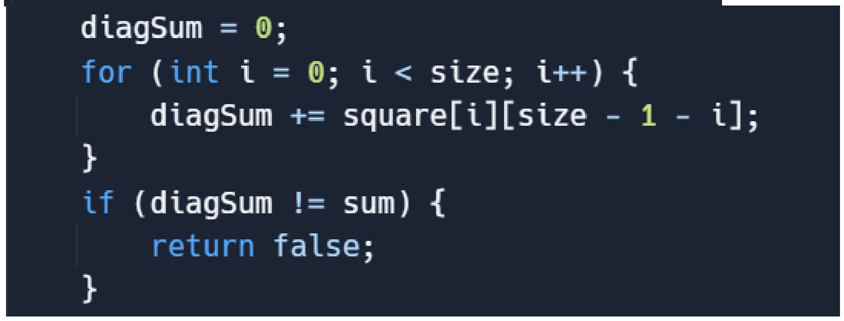


```java
public static boolean magicSquare(int[][] square) {
    int sum = 0;
    int size = square.length;
    int[] allValues = new int[size * size];
    int count = 0;
    for (int i = 0; i < size; i++) {
        sum += square[0][i];
    }

    // check row and column sums
    for (int i = 0; i < size; i++) {
        int rowSum = 0;
        for (int j = 0; j < size; j++) {
            rowSum += square[i][j];
            allValues[count] = square[i][j];
            count++;
        }
        if (sum != rowSum) {
            return false;
        }
    }
```

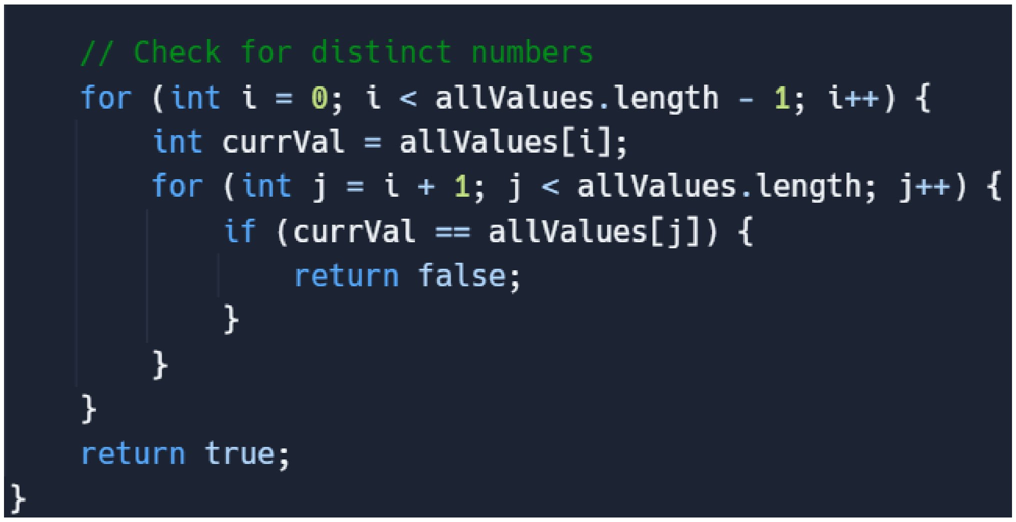
← **magicSquare**



```java
    for (int i = 0; i < size; i++) {
        int colSum = 0;
        for (int j = 0; j < size; j++) {
            colSum += square[j][i];
        }
        if (sum != colSum) {
            return false;
        }
    }

    //check diagonal sums
    int diagSum = 0;
    for (int i = 0; i < size; i++) {
        diagSum += square[i][i];
    }
    if (diagSum != sum) {
        return false;
    }
```



```java
    diagSum = 0;
    for (int i = 0; i < size; i++) {
        diagSum += square[i][size - 1 - i];
    }
    if (diagSum != sum) {
        return false;
    }
```



```java
    // Check for distinct numbers
    for (int i = 0; i < allValues.length - 1; i++) {
        int currVal = allValues[i];
        for (int j = i + 1; j < allValues.length; j++) {
            if (currVal == allValues[j]) {
                return false;
            }
        }
    }
    return true;
}
```