Chapter 7 Problems Part 2

14. Write a method called `contains` that accepts two arrays of integers *a1* and *a2* as parameters and that returns a `boolean` value indicating whether or not the sequence of elements in *a2* appears in *a1* (`true` for yes, `false` for no). The sequence must appear consecutively and in the same order. For example, consider the following arrays:

```
public class ree {
    public static boolean isSubset(int arr1[], int arr2[]) {
        for (int i = 0; i < arr2.length; i++) {
            for (int j = 0; j < arr1.length; j++) {
                if (arr2[i] == arr1[j]) {
                    break;
                }
            }
            if (j == arr1.length) {
                return false;
            }
        }
        return true;
    }
}
```

19. Write a method called `matrixAdd` that accepts a pair of two-dimensional arrays of integers as parameters, treats the arrays as two-dimensional matrices, and returns their sum. The sum of two matrices A and B is a matrix C, where for every row i and column j. $C_{ij} = A_{ij} + B_{ij}$. You may assume that the arrays passed as parameters have the same dimensions.

```
public static int[][] matrixAdd(int[][] arrayA, int[][] arrayB){
        int arrayC[][] = new int[4][4];

        for(int i = 0; i < arrayA.length; i++) {
            for(int j = 0; j< arrayA[0].length; j++) {
                arrayC[i][j] = arrayA[i][j] + arrayB[i][j];
            }
        }
        return arrayC;
    }
```

20. Write a method called `isMagicSquare` that accepts a two-dimensional array of integers as a parameter and returns true if it is a magic

square. A square matrix is a *magic square* if all of its row, column, and diagonal sums are equal.

```
public boolean isMagicSquare(int[][] a) {
    if(a.length == 0)
        return true;
    int sum = 0;
    int len = a[0].length;
    for(int j = 0; j < a[0].length; j++)
        sum += a[0][j];
    for(int i = 1; i < a.length; i++) {
        if(a[i].length != len)
            return false;
        int rowSum = 0;
        for(int j = 0; j < a[i].length; j++)
            rowSum += a[i][j];
        if(rowSum != sum)
            return false;
    }
    for(int j = 0; j < a.length; j++) {
        int colSum = 0;
        for(int i = 0; i < a.length; i++)
            colSum += a[i][j];
        if(colSum != sum)
            return false;
    }
    int diag = 0;
    for(int i = 0; i < a.length; i++)
        diag += a[i][i];
    if(diag != sum)
        return false;
    diag = 0;
    for(int i = 0; i < a.length; i++)
        diag += a[a.length - i - 1][i];
    return diag == sum;
}
```