1.

```
public static int max ( int[][] a ) {
    int max = a[0][0];
    for (int r = 0; r < a.length; r++) {
        for (int c=0; c < a[0].length; c++) {
            if ( a[r][c] > max )
                max = a[r][c];
        }
    }
    return max;
}
```

2.

```
public static int rowSum (int[][] a, int x) {

    int sum = 0;
    int row = x;  // the row will not change

    for (int col = 0; col < a[0].length; col++) {
        sum += a[row][col];
    }

    return sum;
}
```

3.
```
public static int columnSum(int[][] a, int x) {
    int col = x; int sum = 0;
    for (int row = 0; row < a.length; row++) {
        sum += a[row][col];
    }
    return sum;
}
```

4.
```
public static int[] allRowSums(int[][] a) {
    int[] sums = new int[a.length];
    // if the return type is an array, you usually create it.
    for (int r = 0; r < a.length; r++) {
        sums[r] = rowSum(a, r); // parameters are
                                // 2D array, int
    }
    return sums;
}
```

5.
```
public static boolean isRowMagic(int[][] a) {
    // assume true until one rowsum is not equal to 1st one
    int[] rows = allRowSums(a);
    int magicSum = rows[0]; // check that all sums are same
                            // as magicSum
    for (int i=0; i < rows.length; i++) {
        if (rows[i] != magicSum)
            return false;
    }
    return true;
}
```

6.
```
public static boolean isColumnMagic(int[][] a) {
    int[] cols = new int[a.length];
    for (int c = 0; c < a.length; c++) {
        cols[c] = columnSum(a, c);
    }
    int magicSum = cols[0];
    for (int i = 0; i < cols.length; i++) {
        if (cols[i] != magicSum)
            return false;
    }
    return true;
}
```

7.
```
public static boolean isSquare(int[][] a) {
    if (a.length == a[0].length)
        return true;
    return false;
}
```

8. public static boolean isMagic(int[][] a) {

```
boolean magic = isRowMagic(a) && isColumnMagic()
                && isSquare();
// if get past this next if stmt, then 1st three tests good
if (!magic)
    return false;  // good to check 1st bc cannot do diagonal
                   // if not square
int d1sum = 0, d2sum = 0;
for (int r = 0; r < a.length; r++) {
    for (int c = 0; c < a[0].length; c++) {
        if (r == c)
            d1sum += a[r][c];
        if (r + c == a.length - 1)
            d2sum += a[r][c];
    }
if (d1sum == d2sum && d1sum == rowSum(a,0))
    return true;
return false;
}
```