Fill Grid

Here is a video walkthrough of all parts of this problem.

Given two one-dimensional arrays `LL` and `UR`, fill in the program on the next page to insert the elements of `LL` into the lower-left triangle of a square two-dimensional array `S` and `UR` into the upper-right triangle of `S`, without modifying elements along the main diagonal of `S`. You can assume `LL` and `UR` both contain at least enough elements to fill their respective triangles. (Spring 2020 MT1)

For example, consider

```
int[] LL = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 0, 0 };
int[] UR = { 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 };
int[][] S = {
    { 0, 0, 0, 0, 0},
    { 0, 0, 0, 0, 0},
    { 0, 0, 0, 0, 0},
    { 0, 0, 0, 0, 0},
    { 0, 0, 0, 0, 0}
};
```

After calling `fillGrid(LL, UR, S)`, `S` should contain

```
{
  { 0, 11, 12, 13, 14 },
  { 1,  0, 15, 16, 17 },
  { 2,  3,  0, 18, 19 },
  { 4,  5,  6,  0, 20 },
  { 7,  8,  9, 10,  0 }
}
```

(The last two elements of `LL` are excess and therefore ignored.)

```java
/** Fill the lower-left triangle of S with elements of LL and the
 *  upper-right triangle of S with elements of UR (from left-to
 *  right, top-to-bottom in each case). Assumes that S is square and
 *  LL and UR have at least sufficient elements. */
public static void fillGrid(int[] LL, int[] UR, int[][] S) {
    int N = S.length;
    int kL, kR;
    kL = kR = 0;

    for (int i = 0; i < N; i += 1) {

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________

        _____________________________________________
    }
}
```

Solution:

```
public static void fillGrid(int[] LL, int[] UR, int[][] S) {
    int N = S.length;
    int kL, kR;
    kL = kR = 0;
    for (int i = 0; i < N; i += 1) {
        for (int j = 0; j < N; j += 1) {
            if (i < j) {
                S[i][j] = UR[kR];
                kR += 1;
            } else if (i > j) {
                S[i][j] = LL[kL];
                kL += 1;
            }
        }
    }
}
```

Alternate Solutions:

```
public static void fillGrid(int[] LL, int[] UR, int[][] S) {
    int N = S.length;
    int kL, kR;
    kL = kR = 0;
    for (int i = 0; i < N; i += 1) {
        for (int j = 0; j < i; j += 1) {
            S[i][j] = LL[kL];
            kL += 1;
        }
        for (int j = i + 1; j < N; j += 1) {
            S[i][j] = UR[kR];
            kR += 1;
        }
    }
}
```

```
public static void fillGrid(int[] LL, int[] UR, int[][] S) {
    int N = S.length;
    int kL, kR;
    kL = kR = 0;
    for (int i = 0; i < N; i += 1) {
        System.arraycopy(LL, kL, S[i], 0, i);
        System.arraycopy(UR, KR, S[i], i + 1, N - i - 1);
        kL += i;
        kR += square.length - i - 1; */
    }
}
```