Array

Sp18 disc03

```java
public static int[] insert(int[] arr, int item, int position) {

    int[] result = new int[arr.length + 1];
    position = Math.min(arr.length, position);
    for (int i = 0; i < position; i++) {
        result[i] = arr[i];
    }
    result[position] = item;
    for (int i = position; i < arr.length; i++) {
        result[i + 1] = arr[i];
    }
    return result;
}
```

```java
public static void reverse(int[] arr) {

    for (int i = 0; i < arr.length / 2; i++) {
        int j = arr.length - i - 1;
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

number at index `i` with `arr[i]` copies of itself. For example, `replicate([3, 2, 1])` would return `[3, 3, 3, 2, 2, 1]`.

```
public static int[] replicate(int[] arr) {


    int total = 0;
    for (int item : arr) {
        total += item;
    }
    int[] result = new int[total];
    int i = 0;
    for (int item : arr) {
        for (int counter = 0; counter < item; counter++) {
            result[i] = item;
            i++;
        }
    }
    return result;
}
```

Sp18 preExam

For example, `flatten({{1, 2, 3}, {}, {7, 8}})` should return `{1, 2, 3, 7, 8}`. (Summer 2016 MT1)

```
public static int[] flatten(int[][] x) {
    int totalLength = 0;
    for (int i = 0; i < x.length; i++) {
        totalLength += x[i].length;
    }
    int[] a = newint[totalLength];
    int aIndex = 0;
    for (int i = 0; i < x.length; i++) {
        for (int j = 0; j < x[i].length; j++) {
            a[aIndex] = x[i][j];
            aIndex++;
        }
    }
    return a;
}
```

Sp21 preExam

```
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 }
}
```

```
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;
            }
        }
    }
}
```

```java
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;
        }
    }
}
```

```java
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; */
    }
}
```