

```
class MergeSort
```

```
{
```

```
    static void mergesort (int[] x, int p, int r)
```

```
    {
```

```
        // sort the elements x[p], x[p+1], ... , x[r-1]
```

```
        if (p < r-1) // at least two elements to be sorted
```

```
        {
```

```
            int q = (p+r)/2;
```

```
            mergesort (x, p, q);
```

```
            mergesort (x, q, r);
```

```
            merge (x, p, q, r);
```

```
        }
```

```
    }
```

```
    static void merge (int[] x, int p, int q, int r)
```

```
        // merge the two sorted sequences
```

```
        // x[p],... , x[q-1] and x[q], ... , x[r-1]
```

```
    {
```

```
        int[] scratchpad = new int[q-p];
```

```
        // copy the first half to scratchpad
```

```
        for (int i=p; i < q; i++)
```

```
            scratchpad[i-p] = x[i];
```

```
        boolean upperempty = false;
```

```
        int lower = 0; // index to scratchpad
```

```
        int L = scratchpad[lower];
```

```
        int upper = q ; int U = x[upper];
```

```

for (int i=p; i<r ; i++)
{
    if (upperempty) // only rest of bottom half
    {
        x[i] = scratchpad[lower]; lower++;
    }
    else
        if (L <= U)
        {
            x[i] = L; lower++;
            if (lower >= q-p) // scratchpad empty
                break; // all done!
            L = scratchpad[lower];
        }
        else
        {
            x[i] = U; upper++;
            if (upper >= r)
                upperempty = true;
            else
                U = x[upper];
        }
    }
}

```