1. A researcher wishes to calculate some statistical properties for a collection of integer data values. The data values are represented by the array `tally`. The indexes of the array represent the possible values of the actual data values from zero to the maximal value (15 in the example below). Each array location contains the frequency (number of occurrences) of the value corresponding to its index. In the example below, `tally[4]` is 10, which means that the value **4** occurs ten times in the collection of data; whereas `tally[8]` is 0, which means that the value **8** does not occur in the data collection.

`tally`

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Frequency	0	0	10	5	10	0	7	1	0	6	0	10	3	0	0	1

(a) You will write the function `CalculateModes`, which is described as follows. `CalculateModes` returns an array containing the mode(s) found in parameter `tally`. The length of the returned array is equal to the number of modes.

A **mode** is defined as a value that occurs with maximal frequency. If there is more than one such value, each is considered a mode of the data. In the example above, the modes are 2, 4, and 11, because they each occur 10 times and all other values occur fewer than 10 times.

The following function, `FindMax`, is available for your use. It returns the maximum value in array `nums`. Using the example array, `FindMax(tally)` returns 10.

```
int FindMax(const apvector<int> & nums);
// precondition:  nums.length() > 0
// postcondition: returns the maximum value in nums
```

Do NOT write the body of `FindMax`.

In writing `CalculateModes`, you may call `FindMax` as specified above.

Complete function `CalculateModes` below.

```
apvector<int> CalculateModes(const apvector<int> & tally)
// precondition:  tally.length() > 0
// postcondition: returns an apvector that contains the mode(s);
//                the apvector's length equals the number of modes
```

(b) You will write the function `KthDataValue`, which is described as follows. `KthDataValue` returns the kth data value when the data values are considered in sorted order. Recall that the indexes of the array represent possible data values and that each array location contains the frequency of the value corresponding to its index.

In the example reprinted below, the first ten data values are **2**, the next five data values are **3**, and the next ten data values are **4**. `KthDataValue(tally, 1)` returns **2**, `KthDataValue(tally, 14)` returns **3**, `KthDataValue(tally, 15)` returns **3**, and `KthDataValue(tally, 16)` returns **4**.

`tally`

Value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Frequency	0	0	10	5	10	0	7	1	0	6	0	10	3	0	0	1

Complete function `KthDataValue` below.

```
int KthDataValue(const apvector<int> & tally, int k)
// precondition:  tally.length() > 0;
//                0 < k ≤ total number of values in the data collection
// postcondition: returns the kth value in the data collection
//                represented by tally
```