A mode is a value in an array that is larger than both the value immediately before it in the array and the value immediately after it.  In other words, a mode occurs at index k in the array A if A[K] > A[k-1] and A[k] > A[k+1].  The array is unimodal if the values increase until they reach a mode, then decrease, do that there is only one mode.  For example, the array A shown below is unimodal with its mode occurring at index 4.  Assume that the mode does not occur at the first or last entry in the array.

| **Index k** | **A[k]** |
| --- | --- |
| 0 | 3 |
| 1 | 5 |
| 2 | 9 |
| 3 | 10 |
| 4 | 12   <- Mode |
| 5 | 11 |
| 6 | 9 |
| 7 | 4 |

Consider the following class:

class dataSet {

    int data[];

    int size;     //contains the number of items in data

    public dataSet() { /\*constructor - not shown \*/ }

    public boolean isMode(int k) { /\* you will write this in part a \*/}

    public int modeIndex(); { /\* you will write this in part b \*/ }

    pubic void printHistogram(int longestBar, String barchar) {

                        /\* you will write this in part c \*/ }

}

a.  Write the method isMode, as started below.  isMode returns true if data[k] is larger than data[k+1] and larger than data[k-1]; otherwise, it returns false.  In the example above, the call isMode(4) returns true and the call isMode(5) returns false.

Complete the method isMode below.

/\*\*

 \* Before entering the method 0 < k < size - 1

 \*\*/

public boolean isMode(int k) {

b.  Write method modeIndex, as started below.  modeIndex returns the index of the mode of data.  You may assume that data is unimodal and the mode occurs at an index k, where 0 < k < size.  In the example above, the call modeIndex() returns 4.

In writing modeIndex you may call method isMode specified in part (a).  Assume that isMode works as specified, regardless of what you wrote in part (a).

Complete method modeIndex below.

/\*\*

 \* before entering the method data is unimodal and size >= 3

 \*\*/

public int modeIndex() {

c.  Write method printHistogram, as started below.  printHistogram prints a charachter histogram of unimodal array of nonnegative values, data, such that the longest bar of the histogram (the mode) has longestBar charachters barChar, and all other bars have a number of barChar charachters proportional to the corresponding value in the array data (rounding down).

For example, assume that int data[] contains the values shown below.

The call printHistogram(20, "x") will print the histogram shown in the output column below.

| | Index k | data[k] | Length of bar | | --- | --- | --- | | 0 | 3 | 5 | | 1 | 5 | 8 | | 2 | 9 | 15 | | 3 | 10 | 16 | | 4 | 12 | 20 | | 5 | 11 | 18 | | 6 | 9 | 15 | | 7 | 4 | 6 | | Output of call printHistogram(20,"x")  xxxxx  xxxxxxxx  xxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxxxxx  xxxxxxxxxxxxxxx  xxxxxx |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

In writing printHistogram, you may call methods isMode and modeIndex specified in parts (a) and (b).  Assume that isMode and ModeIndex work as specified, regardless of what you wrote in parts (a) and (b).

Complete the method printHistogram below.

/\*\*

 \* before entering this method the data is unimodal and the size >= 3, each data element is greater than 0.

 \*\*/

public void printHistogram(int longestBar, String barChar) {