Complete method `getValueAt` below.

```
/** Returns the value of the element at row index row and column index col in the sparse array.
 *  Precondition:  0 ≤ row < getNumRows()
 *                 0 ≤ col < getNumCols()
 */
public int getValueAt(int row, int col)
```

Part (b) begins on page 13.

GO ON TO THE NEXT PAGE.

(b) Write the `SparseArray` method `removeColumn`. After removing a specified column from a sparse array:

- All entries in the list `entries` with column indexes matching `col` are removed from the list.
- All entries in the list `entries` with column indexes greater than `col` are replaced by entries with column indexes that are decremented by one (moved one column to the left).
- The number of columns in the sparse array is adjusted to reflect the column removed.

The sample object `sparse` from the beginning of the question is repeated for your convenience.

	0	1	2	3	4
0					
1		5			4
2	1				
3		-9			
4					
5					

The shaded entries in `entries`, below, correspond to the shaded column above.

```
numRows: 6

numCols: 5

entries:  row:    1    row:    2    row:    3    row:    1
          col:    4    col:    0    col:    1    col:    1
          value:  4    value:  1    value: -9    value:  5
```

When `sparse` has the state shown above, the call `sparse.removeColumn(1)` could result in `sparse` having the following values in its instance variables (since `entries` is in no particular order, it would be equally valid to reverse the order of its two items). The shaded areas below show the changes.

```
numRows: 6

numCols: 4

entries:  row:    1    row:    2
          col:    3    col:    0
          value:  4    value:  1
```

GO ON TO THE NEXT PAGE.

Class information repeated from the beginning of the question

```
public class SparseArrayEntry

public SparseArrayEntry(int r, int c, int v)
public int getRow()
public int getCol()
public int getValue()

public class SparseArray

private int numRows
private int numCols
private List<SparseArrayEntry> entries
public int getNumRows()
public int getNumCols()
public int getValueAt(int row, int col)
public void removeColumn(int col)
```

Complete method `removeColumn` below.

```
/** Removes the column col from the sparse array.
 *  Precondition: 0 ≤ col < getNumCols()
 */
public void removeColumn(int col)
```

GO ON TO THE NEXT PAGE.

4. This question involves the design of an interface, writing a class that implements the interface, and writing a method that uses the interface.

   (a) A *number group* represents a group of integers defined in some way. It could be empty, or it could contain one or more integers.

   Write an interface named `NumberGroup` that represents a group of integers. The interface should have a single `contains` method that determines if a given integer is in the group. For example, if `group1` is of type `NumberGroup`, and it contains only the two numbers `-5` and `3`, then `group1.contains(-5)` would return `true`, and `group1.contains(2)` would return `false`.

   Write the complete `NumberGroup` interface. It must have exactly one method.

Part (b) begins on page 17.

GO ON TO THE NEXT PAGE.

AP® COMPUTER SCIENCE A
2015 SCORING GUIDELINES

Question 4: Number Group

Part (a)	**Interface: `NumberGroup`**	**2 points**

Intent: *Define interface to represent a number group*

+1 `interface NumberGroup` (*point lost if visibility private*)

+1 `boolean contains(int` *`num`*`);`
(*point lost if visibility not public or extraneous code present*)

Part (b)	**Class: `Range`**	**5 points**

Intent: *Define implementation of* `NumberGroup` *representing a range of numbers*

+1 `class Range implements NumberGroup` (*point lost if visibility private*)

+1 Declares appropriate `private` instance variable(s)

+1 Uses correct constructor header

+1 Initializes instance variables within constructor using parameters
(*point lost if bounds errors occur in container use*)

+1 Computes and returns correct value from `contains`
(*point lost for incorrect method header*)

Part (c)	`contains`	**2 points**

Intent: *Determine whether integer is part of any of the member number groups*

+1 Calls `contains` on elements of `groupList` in context of loop *(no bounds errors)*

+1 Computes and returns correct value

Question-Specific Penalties

-1 (s) Inappropriate use of `static`