

3. A two-dimensional array of integers in which most elements are zero is called a *sparse array*. Because most elements have a value of zero, memory can be saved by storing only the non-zero values along with their row and column indexes. The following complete `SparseArrayEntry` class is used to represent non-zero elements in a sparse array. A `SparseArrayEntry` object cannot be modified after it has been constructed.

```
public class SparseArrayEntry
{
    /** The row index and column index for this entry in the sparse array */
    private int row;
    private int col;

    /** The value of this entry in the sparse array */
    private int value;

    /** Constructs a SparseArrayEntry object that represents a sparse array element
     * with row index r and column index c, containing value v.
     */
    public SparseArrayEntry(int r, int c, int v)
    {
        row = r;
        col = c;
        value = v;
    }

    /** Returns the row index of this sparse array element. */
    public int getRow()
    { return row; }

    /** Returns the column index of this sparse array element. */
    public int getCol()
    { return col; }

    /** Returns the value of this sparse array element. */
    public int getValue()
    { return value; }
}
```

The `SparseArray` class represents a sparse array. It contains a list of `SparseArrayEntry` objects, each of which represents one of the non-zero elements in the array. The entries representing the non-zero elements are stored in the list in no particular order. Each non-zero element is represented by exactly one entry in the list.

```
public class SparseArray
{
    /** The number of rows and columns in the sparse array. */
    private int numRows;
    private int numCols;

    /** The list of entries representing the non-zero elements of the sparse array. Entries are stored in the
     * list in no particular order. Each non-zero element is represented by exactly one entry in the list.
     */
    private List<SparseArrayEntry> entries;

    /** Constructs an empty SparseArray. */
    public SparseArray()
    { entries = new ArrayList<SparseArrayEntry>(); }

    /** Returns the number of rows in the sparse array. */
    public int getNumRows()
    { return numRows; }

    /** Returns the number of columns in the sparse array. */
    public int getNumCols()
    { return numCols; }

    /** 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)
    { /* to be implemented in part (a) */ }

    /** Removes the column col from the sparse array.
     * Precondition: 0 ≤ col < getNumCols()
     */
    public void removeColumn(int col)
    { /* to be implemented in part (b) */ }

    // There may be instance variables, constructors, and methods that are not shown.
}
```

The following table shows an example of a two-dimensional sparse array. Empty cells in the table indicate zero values.

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

The sample array can be represented by a `SparseArray` object, `sparse`, with the following instance variable values. The items in `entries` are in no particular order; one possible ordering is shown below.

numCols: 5													
entries:													
	<table><tr><td>row: 1</td><td>row: 2</td><td>row: 3</td><td>row: 1</td></tr><tr><td>col: 4</td><td>col: 0</td><td>col: 1</td><td>col: 1</td></tr><tr><td>value: 4</td><td>value: 1</td><td>value: -9</td><td>value: 5</td></tr></table>	row: 1	row: 2	row: 3	row: 1	col: 4	col: 0	col: 1	col: 1	value: 4	value: 1	value: -9	value: 5
	row: 1	row: 2	row: 3	row: 1									
	col: 4	col: 0	col: 1	col: 1									
value: 4	value: 1	value: -9	value: 5										

(a) Write the `SparseArray` method `getValueAt`. The method returns the value of the sparse array element at a given row and column in the sparse array. If the list `entries` contains an entry with the specified row and column, the value associated with the entry is returned. If there is no entry in `entries` corresponding to the specified row and column, 0 is returned.

In the example above, the call `sparse.getValueAt(3, 1)` would return -9, and `sparse.getValueAt(3, 3)` would return 0.

WRITE YOUR SOLUTION ON THE NEXT PAGE.

Part (a) continues on page 12.

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.

(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:	<table border="1"> <tr> <td>row: 1 col: 4 value: 4</td> <td>row: 2 col: 0 value: 1</td> <td>row: 3 col: 1 value: -9</td> <td>row: 1 col: 1 value: 5</td> </tr> </table>	row: 1 col: 4 value: 4	row: 2 col: 0 value: 1	row: 3 col: 1 value: -9	row: 1 col: 1 value: 5
row: 1 col: 4 value: 4	row: 2 col: 0 value: 1	row: 3 col: 1 value: -9	row: 1 col: 1 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:	<table><tr><td>row: 1 col: 3 value: 4</td><td>row: 2 col: 0 value: 1</td></tr></table>	row: 1 col: 3 value: 4	row: 2 col: 0 value: 1
row: 1 col: 3 value: 4	row: 2 col: 0 value: 1		

```
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)
```