

Question 1

25 minutes to complete, 5 minutes to upload answer.

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:  $0 \leq \text{row} < \text{getNumRows}()$ 
     *  $0 \leq \text{col} < \text{getNumCols}()$ 
     */
    public int getValueAt(int row, int col)
    { /* to be implemented in part (a) */ }

    /** Returns an integer array containing each value from each row sequentially (including 0 values),
     * starting at the first value of the top row, and ending at the last value of the bottom row.
     */
    public int[] serialize()
    { /* 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.

	5			4
1				
	-9			

(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.

```
numRows: 6
numCols: 5
```

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

(b) Write the `SparseArray` method `serialize`. The method returns a one-dimensional integer array containing all the values (including 0 values) from the array, starting at the first value of the top row, and ending at the last value of the bottom row.

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)

{
```

```
/** Returns an integer array containing each value from each row sequentially (including 0 values),  
 * starting at the first value of the top row, and ending at the last value of the bottom row.  
 */
```

```
public int[] serialize()
```

```
{
```