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.
}
```

AP® COMPUTER SCIENCE A
2015 SCORING GUIDELINES

Question 3: Sparse Array

Part (a)	`getValueAt`	**3 points**

Intent: *Return the value at row index* `row` *and column index* `col` *in sparse array*

+1 Accesses all necessary elements of `entries` (*No bounds errors*)

+1 Identifies element of `entries` at row index `row` and column index `col`, if exists

+1 Returns identified value or returns 0 if no entry exists in `entries` with row index `row` and column index `col`

Part (b)	`removeColumn`	**6 points**

Intent: *Remove column* `col` *from sparse array*

+1 Decrements `numCols` exactly once

+1 Accesses all elements of `entries` (*No bounds errors*)

+1 Computes sum of row in `arr2D` using `arraySum` and assigns to element in 1D array

+1 Identifies and removes entry with column index `col`

+2 Process entries with column index `> col` within loop

- **+1** Creates new `SparseArrayEntry` with current row index, column index -1, current value
- **+1** Identifies and replaces entry with column index `> col` with created entry

+1 On exit: All and only entries with column index `col` have been removed and all and only entries with column index `> col` have been changed to have column index -1. All other entries are unchanged. (*Minor loop errors ok*)

Question-Specific Penalties

-2 (t) Consistently uses incorrect name instead of `entries`

-1 (u) Directly accesses private instance variables in `SparseArrayEntry` object