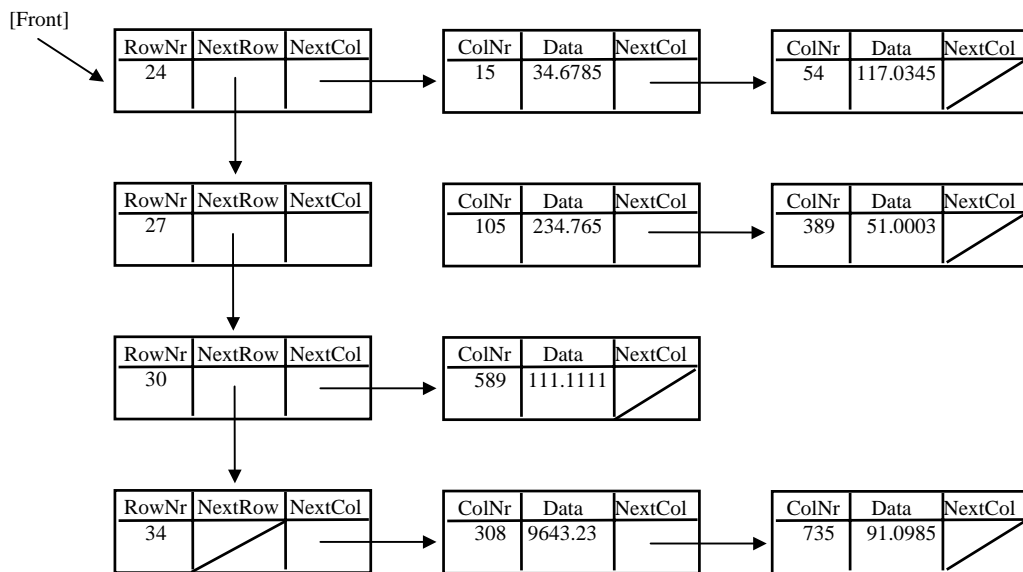


### Assignment Purpose:

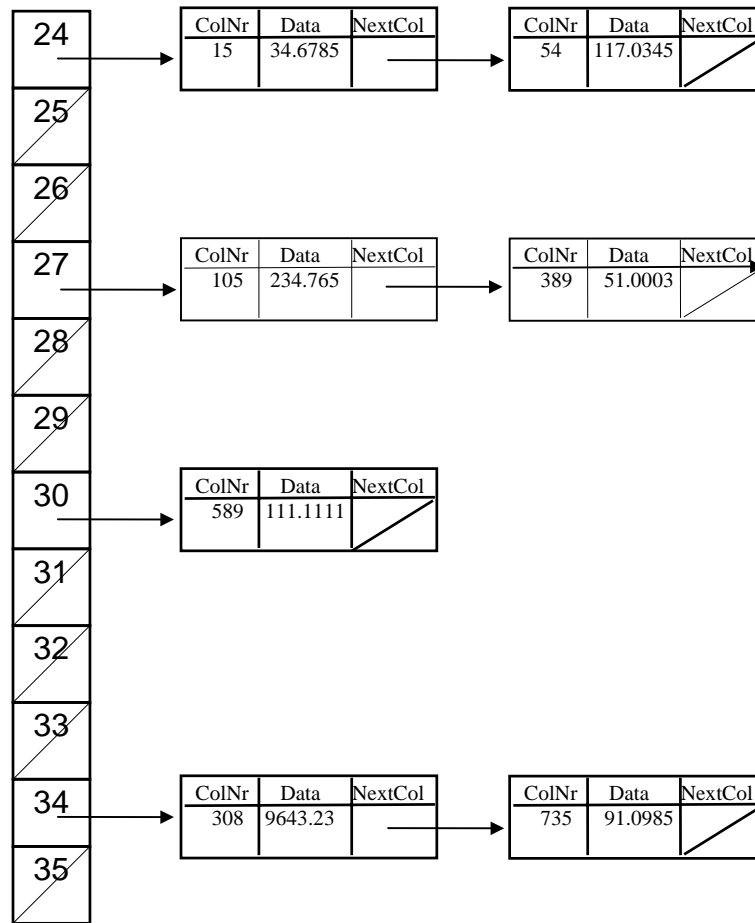
The purpose of this program is to demonstrate knowledge of complex dynamically linked data structures. This program requires understanding of one-dimensional and two-dimensional linked list data structures.

Write a program that performs the exact same data processing as program `SparseProg.java` and `ColNode.java` and `RowNode.java` in the links below. This demonstration program at the end of chapter 34 implements a sparse matrix program with a *linked list of linked lists*. You need to alter this program and implement the sparse matrix data structure with an *array of linked lists*.

The program in chapter 34 is implemented with a linking structure that can be represented by the following illustration. Note that there are special header nodes that start each individual linked list. Only the row nodes store the row value. All other nodes are concerned with storing the column value.



The change to *array of linked lists* is illustrated by the diagram on the next page. It represents the same sparse matrix information as the previous drawing. The only difference is the implementation. The row value of each sparse matrix value is the index of the array.



You will start this assignment with program `SparseProg.java`. In other words the **Lab34st.java** file is in fact the same program as `SparseProg.java`. Do not start from scratch. Examine the student file with its *linked list of linked lists* implementation and make the necessary changes so that your sparse matrix is now implemented with an *array of linked lists*. The student file is shown next, but it should not be necessary to copy the long value. The electronic file will be provided for you.

```
// This is the student version of the Lab34 assignment. This program implements a "sparse matrix" class using a linked list of linked lists.
import java.io.*;

public class SparseProg
{
    public static void main (String args[]) throws IOException
    {
        Sparse sparse = new Sparse();
        sparse.enterSparse();
        sparse.displaySparse();
        System.out.println("\n\n");
    }
}
```

```

class Sparse
{
    private RowNode head;    // points to the header node of the first linked list
    private boolean first;   // determines if first node is created or not

    public Sparse()
    {
        head = null;
        first = true;
    }

    public void enterSparse() throws IOException
    {
        BufferedReader input = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("\nEnter [ -999 <cr> ] after last data entry to exit\n");
        boolean done = false;
        do
        {
            System.out.println();
            System.out.print("Enter row number ===>> ");
            int row = Integer.parseInt(input.readLine());
            done = row == -999;
            if (!done)
            {
                System.out.print("Enter col number ===>> ");
                int col = Integer.parseInt(input.readLine());
                System.out.print("Enter sparse value ===>> ");
                int val = Integer.parseInt(input.readLine());
                if (first)
                {
                    firstNode(val,row,col);
                    first = false;
                }
                else
                {
                    insertNode(val,row,col);
                }
            }
        }
        while (!done);
    }

    public void firstNode(int value, int row, int col)
    {
        ColNode temp = new ColNode(value,col,null);
        head = new RowNode(row,temp,null);
    }

    public void insertNode(int value, int row, int col)
    {
        ColNode c1 = null, c2 = null, c3 = null;    // used to create new col nodes
        RowNode r1 = null, r2 = null, r3 = null;    // used to create new row nodes

        // create new col node with new value
        c1 = new ColNode(value,col,null);

        if (row < head.getRowNumber())
            // create new first header node if row is less than current first row
            {
                r1 = new RowNode(row,c1,head);
                head = r1;
            }
        else
        {
            r2 = head;
            while (r2 != null && row > r2.getRowNumber())
                // traverse to find the proper row location
                {
                    r3 = r2;
                    r2 = r2.getNextRow();
                }
            if (r2 != null && row == r2.getRowNumber())
                // new node is inserted in an existing row
                {
                    c2 = r2.getNextCol();
                    if (col < c2.getColNumber())

```

```

        // new node fits between header node and the first col node
        {
            r2.setNextCol(c1);
            c1.setNextCol(c2);
        }
        else
        {
            while (c2 != null && col > c2.getColNumber())
                // find proper col insertion position
                {
                    c3 = c2;
                    c2 = c2.getNextCol();
                }
            c3.setNextCol(c1);
            c1.setNextCol(c2);
        }
    }
    else
    {
        r1 = new RowNode(row,c1,r2);
        r3.setNextRow(r1);
    }
}

public void displaySparse()
{
    System.out.println();
    RowNode r = head;
    ColNode c = null;
    while (r != null)
    {
        c = r.getNextCol();
        while (c != null)
        {
            int row = r.getRowNumber();
            int col = c.getColNumber();
            int val = c.getValue();
            System.out.println "[" + row + "," + col + "] \t= " + val);
            c = c.getNextCol();
        }
        r = r.getNextRow();
    }
}

class ColNode
{
    public ColNode (int initValue, int initColNumber, ColNode initNextCol)
    {
        value = initValue;
        colNumber = initColNumber;
        nextCol = initNextCol;
    }

    public int getValue ()           { return value;           }
    public int getColNumber ()       { return colNumber;     }
    public ColNode getNextCol ()     { return nextCol;         }
    public void setValue (int theNewValue) { value = theNewValue; }
    public void setNextCol (ColNode theNewNextCol) { nextCol = theNewNextCol; }

    private int colNumber;
    private ColNode nextCol;
    private int value;
}

class RowNode
{

```

```

public RowNode (int initRowNumber, ColNode initNextCol, RowNode initNextRow)
{
    rowNumber = initRowNumber;
    nextCol = initNextCol;
    nextRow = initNextRow;
}

public int getRowNumber ()           { return rowNumber;      }
public ColNode getNextCol ()        { return nextCol;      }
public RowNode getNextRow ()        { return nextRow;      }
public void setNextCol (ColNode theNewNextCol) { nextCol = theNewNextCol; }
public void setNextRow (RowNode theNewNextRow) { nextRow = theNewNextRow; }

private int rowNumber;
private ColNode nextCol;
private RowNode nextRow;
}

```

## Lab34 100 Point Version

## One Required Output

Enter [ -999 <cr> ] after last data entry to exit

```

Enter row number   ====> 137
Enter col number   ====> 342
Enter sparse value ====> 1000

Enter row number   ====> 101
Enter col number   ====> 698
Enter sparse value ====> 2000

Enter row number   ====> 892
Enter col number   ====> 112
Enter sparse value ====> 3000

Enter row number   ====> 101
Enter col number   ====> 502
Enter sparse value ====> 4000

Enter row number   ====> 500
Enter col number   ====> 400
Enter sparse value ====> 5000

Enter row number   ====> 137
Enter col number   ====> 100
Enter sparse value ====> 6000

Enter row number   ====> -999

[101,502]         = 4000
[101,698]         = 2000
[137,100]         = 6000
[137,342]         = 1000
[500,400]         = 5000
[892,112]         = 3000

```