Data Structure Homework 2 (15+2 pts)

- Due Time: November 12th, 23:59PM
- Late submission will NOT get credits.
- But you can submit parts of the project to get partial credits.
- If you cannot finish all, submit solutions for some problems to get partial credits.

About Submission (Please read carefully, otherwise you may lose points).

- Only submit source files. Do NOT include any executables. All files should be saved in a folder and then packed into a single .zip file and be submitted via blackboard (NOT .rar or .tar.gz).
- The folder name (before compression) as well as the final zip file name should be "FirstName-LastName-HW2".
- Ensure your code can be compiled and executed in command line (not a java IDE). Otherwise, you will NOT get any credits.
- In the zip file, include a text file: README.txt. Write down which problems have you finished. Also write down on which platform (mac, linux, window) is the code compiled and executed.
- You are NOT allowed to use any native implementation of lists (ArrayList, LinkedList, etc.). Consult the instructor if you want to use any native class that is not mentioned here.
- This is NOT something you can finish in three days. To understand the problem itself takes quite some time. You have to start as early as possible.

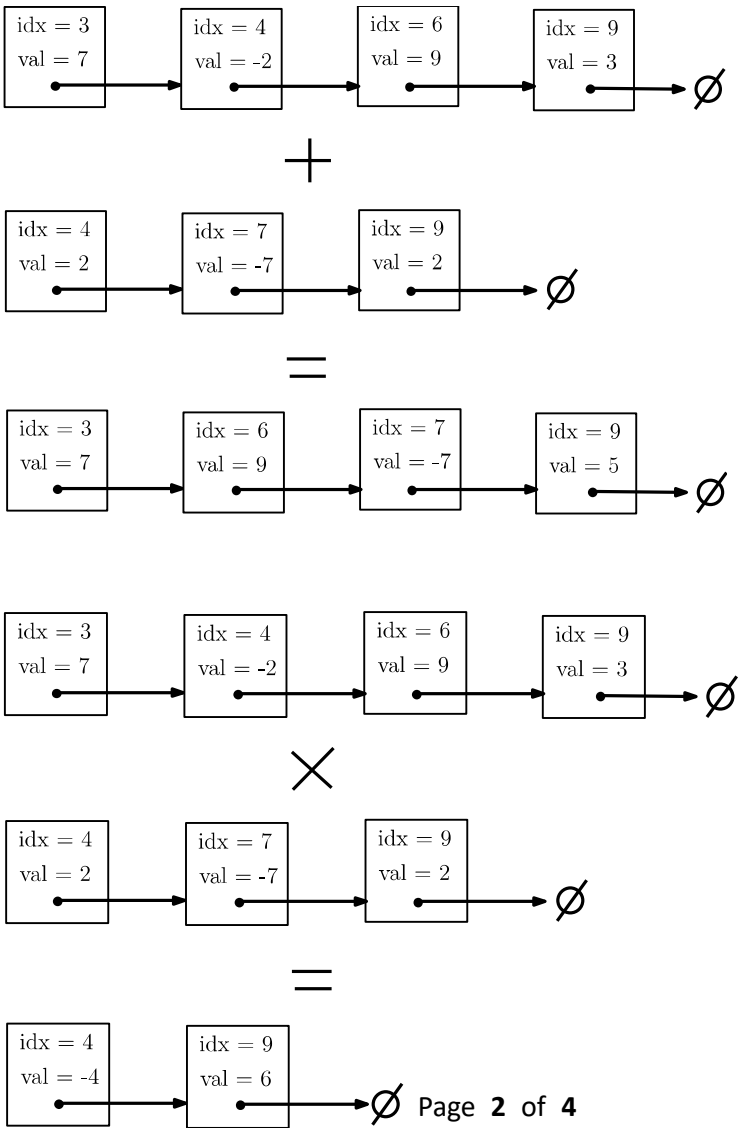
LLMainClass.java and the interface files (SparseM and SparseVec) should not be changed when you submit. That is, you should not change the name/input/output of these major functions.

Problem 1: Sparse Vector Using Linked List (5 pts)

- Implement the linked list sparse vector class (LLSparseVec.java) so that LLMainClass can be executed.
- Nodes in the linked list are nonzero elements of the vector, sorted according to their indices. The length of a vector is specified at construction.
- When an element is set to zero, the corresponding node should be removed!
- Implement the constructor, accessor methods, getElement, setElement, clearElement, getAllIndices, getAllValues. In otherwords, when LLMainClass is called using VEC argument and with a single input file, the program should be able to run correctly and give the same output as ArrayMainClass.

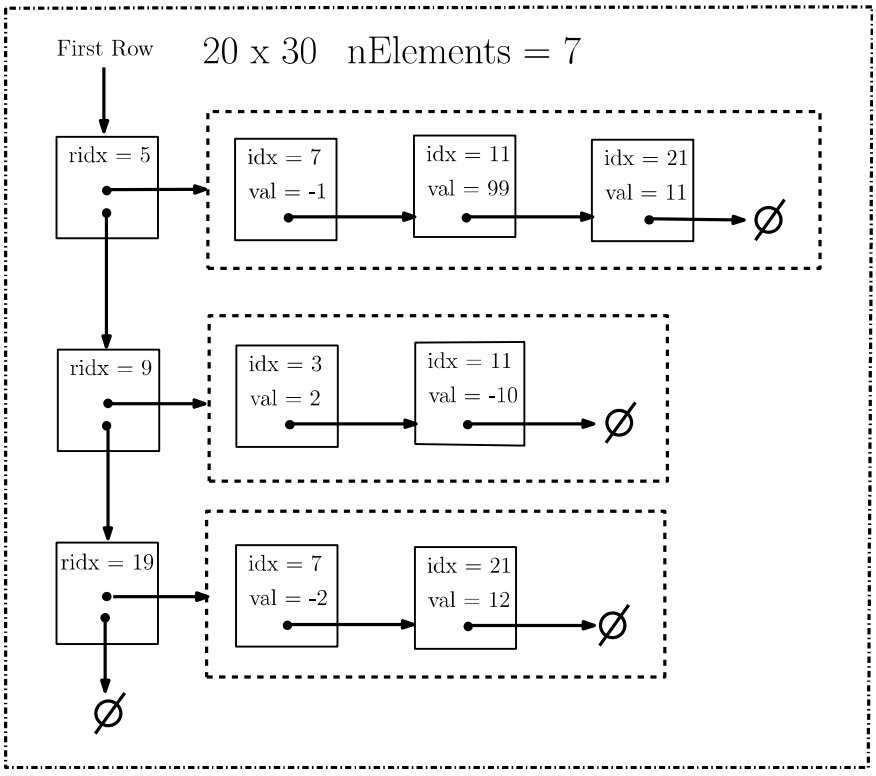
- Implement both the addition, subtraction and multiplication methods in LLSparseVec. The method subtraction(otherV) means the current vector minus otherV.
- The algorithm has to be O(m), in which m is the maximum number of nonzero elements in a vector (length of the list). To achieve this, you cannot simply use get and set in Problem 1. **Only algorithms with O(m) complexity will get credits.** The smart-merge method in HW1 is a good place to start. But note the difference here.
- All operations return a new sparse vector object, storing the result.
- If the two vectors' length do not match, return a null object.
- When LLMainClass is called using VEC argument and with multiple input files, the program should be able to run correctly and give the same output as ArrayMainClass.

Examples:



Problem 2: Sparse Matrix Using Linked List (5 pts)

- Implement the linked list sparse matrix class (LLSparseMat.java) so that LLMainClass can be executed with MAT argument.
- RowHead nodes correspond to nonzero rows. Each rowhead node stores a LLSparseVec, storing a nonzero row. It also has a pointer to the next rowhead.
- When a row becomes empty (no nonzero elements), the rowhead should be removed.
- Implement the constructor, accessor methods:
  - getElement, setElement, clearElement, numElements (returns number of non-zero elements).
  - getRowIndices returns an array of indices of rows with nonzero elements.
  - getOneRowColIndices returns an array of nonzero column indices of the row. Use LLSparseVec.getAllIndices().
  - getOneRowValues returns an array of nonzero values. Use LLSparseVec.getAllValues().
  - NOTE that these methods should all be linear to the number of nonzero rows or nonzero elements.
- Sanity check: when LLMainClass is called using MAT argument and with a single input file, the program should be able to run correctly and give the same output as ArrayMainClass.



Problem 3: Sparse Matrix Operation Linked List (5 points)

- Implement the addition, subtraction and multiplication methods in LLSparseM.
- The algorithm has to be O(m), in which m is the maximum number of nonzero elements in a matrix. To achieve this, you cannot simply use get and set in Problem 3. Only algorithms with O(m) complexity will get credits.
- All operations return a new sparse matrix object, storing the result. The method subtraction(otherM) means the current vector minus otherM.
- If the two matrices' dimensions (nrows, ncols) do not match, return a null object.
- When LLMainClass is called using MAT argument and with multiple input files, the program should be able to run correctly and give the same output as ArrayMainClass.

Problem 4: Performance Evaluation (2 bonus points)

Similar to HW1, write a report on matrix construction and operation time, comparing Array and LL implementation: 2 plots, on construction, and on operations. Each plot has two curves: Array implementation, and LL implementation. X-axis is the different sizes of these matrices.

Samples:

In Data folder, there are example inputs. At some stage, you may want to try out more advanced results. Use content in ManyTestCases to further test your program.