CSCE 221 Cover Page
Programming Assignment #3
Due March 12 at midnight to CSNet

Jessica Fang UIN: 224003796

User Name: sndnzki E-mail address: jessicafang@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People	Ryan Yantz	Alex Park			
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Jessica Fang Date 03/06/2015

a) This program works with sparse matrices. The program is able to create matrices from and perform elementary operations on them. The purpose of this assignment is to utilize our knowledge of data structures and our previous vector and doubly linked list classes to create a sparse matrix and estimate the time and space compl of the functions within the vector class.

b) In completing this assignment, I learned about the implementations of doubly linked lists, in particular, sparse matrices.

DoublyLinkedList(const DoublyLinkedList<T>& dll)
runtime: O(n)
DoublyLinkedList<T>& operator=(const DoublyLinkedList<T>& dll)
runtime: O(n)
ostream& operator<<(ostream& out, const DoublyLinkedList<T>& dll)
runtime: O(n)
void resize(). reset the vector
runtime: O(1) space: O(1)
T* getElemRef(). return the reference of an element
runtime: O(1) space: O(1)
void setVal(double n). sets the data value inside an Element
runtime: O(1) space: O(1)
SparseMatrix(int m, int n). creates a matrix with m rows and n columns
runtime: O(n) space: O(1)
SparseMatrix(const SparseMatrix& input). creates a copy of an existing matrix
runtime: O(n) space: O(1)
SparseMatrix& SparseMatrix::operator=(const SparseMatrix& input). sets a matrix equal to another matrix (makes a copy)
runtime: O(n) space: O(1)
void set_rows(int n). sets the number of rows in the matrix
runtime: O(n) space: O(n)
void set_cols(int n). sets the number of rows in the matrix
runtime: O(1) space: O(1)
double get_element_at(int i, int j) const. returns element at (i,j)
runtime: O(n) space: O(1)
void set_element_at(int i, int j, double n). inserts or replaces element at (i,j)
runtime: O(n) space: O(1)
double& operator()(int i, int j). returns a reference to the element at (i,j)
runtime: O(n) space: O(1)
double operator()(int i, int j) const. returns the value of the element at (i,j)
runtime: O(n) space: O(1)
SparseMatrix& operator+(const SparseMatrix& input) const. returns result matrix of the addition of two matrices
runtime: O(n^2) space: O(1)
SparseMatrix& operator-(const SparseMatrix& input) const. returns result matrix of the subtraction of two matrices
runtime: O(n^2) space: O(1)
DenseVector& operator*(const DenseVector& input) const. returns result matrix of the multiplication of an m x n matrix and a n x 1 column vector.
runtime: O(n^2) space: O(1)
SparseMatrix& operator*(const SparseMatrix& input) const. returns result matrix of the addition of two matrices

runtime: O(n^3) space: O(1)

void transpose(). transposes the matrix

runtime: O(n^2) space: O(1)

void consolidate(). deletes all Elements containing a "0" value in the matrix data structure

runtime: O(n^2) space: O(1)

void print(const string &title). prints out the matrix in the form of a matrix

runtime: O(n^2) space: O(1)

ostream& operator<<(std::ostream &os, const SparseMatrix &A). prints out matrix in the form of row#, col #, value

runtime: O(n^2) space: O(n)

istream& operator>>(std::ostream &is, SparseMatrix &A). takes in input stream and creates a matrix

runtime: O(n) space: O(1)

Functions with a runtime of O(1) are the same time complexity as a simple dense vector (n x 1 matrix) whereas functions with complexities of O(n) or higher will take much more time to complete when called on a matrix as compared to being called on a simple dense vector (n x 1 matrix)

c) How to compile: use the makefile (i.e. type "make" into the command line). How to run: ./main.

d) A Range exception is thrown when the rank given in the insert, replace, and remove functions are either less than 0 or greater than the size of the vector. A Dimension exception is thrown when the number of rows and / or columns do not match during matrix addition and subtraction or the number of columns of the first vector does not match the number of rows in the second vector during matrix multiplication.

e) Templates are used in the My_vec and TemplateDoublyLinkedList ADTs in the program.

f) Valid Cases: Doubles inserted Elements into Doubly Linked Lists into existing My_vec rows and existing columns (0 <= n < total rows / total columns)

Invalid Cases: Input other than doubles or input inserted outside existing rows and/or columns (n < 0 or n => total rows / total columns)

Random Cases: none.