COP 4530: Summer 2023
Practice Assignment (Homework 0)

Total Points: 50
Due: Monday 06/05/2023 11:59:00 PM through Linprog

1 Objective

This is a review assignment for COP 3330 concepts. By completing this assignment you will be able to

- Work with templated classes.
- Add functionality to an existing implementation of a standard abstract data structure
- Use the data structure to solve a simple problem.
- Understand Change Propagation

2 Specifications

For this assignment, you will be working with the List class we have already discussed. The class implements a Linked list. The files you will be modifying are available on Top Hat.

You will modify the List class to provide support for Sparse Matrices. A Sparse Matrix is a large matrix with only a small set of non-zero values. An example is a road connectivity graph represented as a matrix. In order to save on memory and time, sparse matrices may be stored as Linked Lists.

For this assignment, you will be modifying the List class to include the following functionality:

1. Download the `list.h` and `list.hpp` files from the Top Hat folder for Friday, 5/26.
2. Add a comment on each file with your name and FSUId. (3 points)
3. Modify the `Node` class to add an integer data member for the column number. (10 points)
    - This would require a getter and setter in the `Node` class as well.
    - Then, identify all the existing functions that have to be modified to accommodate this change and make the required changes.
    - When you modify the `print` function, print the column and the value comma separated followed by a tab, as shown in the sample.
4. Add an operator overload for the + operator for the `List` class. Follow the established convention for the + operator. This will implement sparse matrix addition. For matrix addition, the value in a spot in the result matrix is the sum of the values (or concatenation for strings) in the corresponding spots for the two operand matrices. (10 points)
5. Modify the extraction operator overload to only read in valid (not-null) values. (10 points)

- For this function, you may assume the first entry will be the null-value for the type, which will not be a part of the row but will be used for comparison for null-value entries.
- The rest of the values entered would be the row.
- Count the values as they come in. Only enter the not-null values into the row. The counts (starting from 0) would be the column number for the node.
- Look at the sample run for more details.

6. Create a driver file for this modified `List` class. Call this file `matrix.cpp` (2 points)
7. Write a function called `sparseTranspose` in the driver file. This function will accept an array of Lists as a parameter and return an array of Lists. The function will return the transpose of the matrix that is accepted as the parameter. The number of rows on the transposed matrix would be the largest column number among all the rows in the array. (10 points)
8. In the main function, instantiate the Sparse Matrix for integers, doubles, and strings. Then demonstrate the functionality as described below. (15 points, 5 points for each kind).
   - A Sparse Matrix as a dynamic array of List objects. Each List object would be 1 row of the matrix.
   - Read in the number of rows and create the Matrix. Then, read in the values from the user
   - Instantiate and read in 2 of each kind of matrix. Then perform the addition and transpose operations for each. Print as shown in the sample run.

3 Sample Run

```
Sparse Integer Matrix:
Enter the number of rows: 5
Enter the values 1 row at a time. The first value will be the 0-value:
0 0 15 0 0 9 0 4
0 11 0 0 0 0 0 0
0 0 0 0 12 0 0 3
0 -6 0 0 3 0 0 0
0 0 0 0 0 1 0 0

The Sparse Matrix is:
Row 0: 1, 15    4, 9    6, 4
Row 1: 0, 11
Row 2: 3, 12    6, 3
Row 3: 0, -6    3, 3
Row 4: 4, 1

Enter the number of rows: 5
Enter the values 1 row at a time. The first value will be the 0-value
0 0 0 0 0 0 0 0
0 1 2 3 0 0 0 0
0 0 4 0 7 0 0 0
0 13 0 0 0 6 0 0
0 0 0 0 6 0 0 0

The Sparse Matrix is:
Row 0:
```

```
Row 1: 0, 1 1, 2    2, 3
Row 2: 1, 4    3, 7
Row 3: 0, 13    4, 6
Row 4: 3, 6

The sum is
Row 0: 1, 15    4, 9    6, 4
Row 1: 0, 12    1, 2    2, 3
Row 2: 1, 4    3, 19    6, 3
Row 3: 0, 7 3, 3    4, 6
Row 4: 3, 6 4, 1

The transpose of the first matrix is
Row 0: 1, 11    3, -6
Row 1: 0, 15
Row 2:
Row 3: 2, 12    3, 3
Row 4: 0, 9 4, 1
Row 5:
Row 6: 0, 4 2, 3

The transpose of the second matrix is
Row 0: 1, 1 3, 13
Row 1: 1, 2 2, 4
Row 2: 1, 3
Row 3: 2, 7 4, 6
Row 4: 3, 6
```

4 Submission Instructions

You will be submitting the file directly through the programming server. Please follow these instructions:

1. Please create a tarball called `FSUID_sparseMatrix.tar`, where FSUID is your Canvas login, with the following files:
   - list.h
   - list.hpp
   - matrix.cpp
2. When you're ready to submit, first change the file permissions to give us read permissions: `chmod 704 FSUID_sparseMatrix.tar`
3. Run the submission script: `~jayarama/homework/submitHW0 FSUID_sparseMatrix.tar`

Please note the following:

1. Do not copy-paste the submission script. PDF's use Unicode characters that won't copy correctly on your terminal. Type it in.
2. THE FILE HAS TO BE NAMED CORRECTLY. You run the risk of overwriting other files
3. WE ONLY NEED YOUR TAR FILE. This is the file that contains your solution. Please don't submit your object file or the executable.

4. IF YOU GET THE "File Submitted" MESSAGE, YOU HAVE SUCCESSFULLY SUBMITTED THE FILE.

5. IF YOU MAKE MULTIPLE SUBMISSIONS, THE OLDER SUBMISSIONS WILL BE DELETED. The script will only keep the latest version of your file, and that is the submission we'll grade.

5 General Guidelines

1. Please include a line with your name and your FSUID in each file you turn in.

2. **This is individual work. You may NOT collaborate with other students in the course, former students, hire tutors to "help", copy solutions off the internet, or use pay-for solution websites, including but not limited to Chegg, CourseHero, WiseAnt, Bartelby, WizePrep, tutor.com, assorted Social Media groups, Discord servers where people offer "100% plagiarism-free solutions", any GroupMe groups students might have created, etc.) This includes posting the problem statements on these websites even if you do not use the answer obtained. Doing so is a violation of the Academic Honor Code. Violation of the Honor Code will result in a 0 grade on the program, a reduced letter grade in the course and potentially more serious consequences.**

3. **While using assistive technologies to generate your code might seem like a smart thing to do right now, you will eventually find yourself in a difficult position of being unable to pass the course due to failing the test average requirement**. At this level, it is recommended you write the program yourself to truly understand how programming works. Also, this would violate the Honor Policy.

4. Please make sure you're only using the concepts already discussed in class. Please do not use the STL classes, iterators, etc.

5. If we have listed a specification and allocated point for it, you will lose points if that particular item is missing from your code, even if it is trivial.

6. No global variables (variables outside of main()). constants and symbolic constants are allowed.

7. No `goto` statements.

8. All input and output must be done with streams, using the library `iostream`.

9. Please make sure that you're conforming to specifications (program name, function names, parameters and return types, print statements, expected inputs and outputs etc.). Not doing so will result in a loss of 10 points. Please note that names are case sensitive.

10. Try not to use `break` or `continue` statements in loops. break is allowed only for preventing the fall-though in a `switch` statement. Unnecessary and frivolous use of these would result in a loss of 10 points per statement.

11. NO C style printing is permitted. (Aka, don't use printf). Use cout if you need to print to the screen.

12. When you write source code, it should be readable and well-documented (comments).

13. Make sure you either develop with or test with linprog (to be sure it reports no compile errors or warnings) before you submit the program.

14. Testing your program thoroughly is a part of writing good code. We give you sample runs to make sure you match our output requirements and to get a general idea of how we would test your code. Matching your outputs for JUST the sample runs is not a guarantee of a 100. We have several extensive test cases.

15. Please make sure you've compiled and run your program before you turn it in. Compilation errors can be quite costly. We take 5 points off per compiler error for the first 9 errors. The 10th compiler error will result in a grade of 0.

16. Only a file turned in through the submission script counts as a submission. A file on your computer or linprog, even if it hasn't been edited after the deadline, does not count.

17. The student is responsible for making sure they have turned in the right file(s). We will not accept any excuses about inadvertently modifying or deleting files, or turning in the wrong files.

18. **Program submissions** should be done through the linprog submission script (if it's not there yet I'll create it soon.) Do not send program submissions through e-mail – e-mail attachments will not be accepted as valid submissions.

19. The ONLY files you will submit via linprog is `spraseMatrix.tar`

20. **General Advice** - always keep an untouched copy of your finished homework files in your email. These files will have a time-stamp which will show when they were last worked on and will serve as a backup in case you ever have legitimate problems with submitting files through linprog. Do this for ALL programs.