CS 201, Summer 2018
Homework Assignment 3

Due: 23:55, July 20, 2018

In this homework assignment, you are supposed to implement a scientific paper repository that contains multiple papers by using linked lists. In this paper repository, for each paper, you will have an entry in which you keep:

1. The paper title,
2. The journal, in which the paper is/was published
3. The year the paper is/was published, and
4. A list of its coauthors. For each coauthor, you should keep:
    (a) The first and last name of the coauthor and
    (b) The institution of the coauthor.

In your implementation, for a paper repository, you **MUST** keep its paper entries in a circular doubly linked list. For each of these paper entries, you **MUST** use another linked list to keep its coauthors; this **MUST** be a sorted linked list where the coauthors **MUST** be ordered alphabetically (first according to last name, then according to first name).

Your system will have the following functionalities; the details of these functionalities are given below:

1. Add a paper
2. Remove a paper
3. Show the list of papers
4. Add a coauthor to a paper
5. Remove a coauthor from all papers
6. Query a coauthor
7. Query the papers which are/were published in a particular journal

PART A:

To take the final exam, you MUST submit at least this part and MUST get at least half of its points.

This part includes the following three functionalities:

1. Add a paper
2. Remove a paper
3. Show the list of papers

PART B:

In this part, you will complete the implementation of the entire scientific paper repository system (all of the 7 funcitonalities listed above).

The details of all of the functionalities are given below:

Add a paper: This function adds an entry to the repository for a given paper whose title, journal, and publication year are specified as parameters. In this function, the coauthor list is not specified; the coauthors of the paper will be added later. In this system, the titles of the papers are unique. Thus, if the user attempts to add a paper with an existing title, you should not allow this operation and give a warning message.

Remove a paper: This function removes a paper entry, whose title is specified as a parameter, from the repository. If this paper does not exist in the repository (i.e., if there is no paper with the specified title), you should not allow this operation and give a warning message.

Add a coauthor to a paper: This function adds a coauthor to the coauthor list of a paper. For that, the title of the paper to which the coauthor is added, the coauthor name (first name + last name), and the institution of the coauthor are specified as parameters. In this function, you should consider the following issues:

- If the paper with a specified title does not exist in the repository, you should not allow the operation and give a warning message.
- In this system, all coauthor names are unique within the same paper. Thus, if the user attempts to add a coauthor with an existing name (to the same paper), you should not perform the operation and give a warning message. Additionally, a coauthor can be affiliated to different institutions in different papers; but the coauthor cannot be affiliated to different institutions in the same paper.
- The coauthor list should be in alphabetically ascending order according to the last names of coauthor; if you have more than one entry with the same last name, you should also consider the first names.

Remove a coauthor from all papers: This function removes a coauthor from all the paper that this coauthor has written. For that, the coauthor name (first name + last name) is specified as a parameter. If the coauthor is not in the coauthor list of any paper, you should not allow the operation and give a warning message. Note that the coauthor list should remain sorted in ascending order after calling this function.

Show the list of papers: You should list all paper entries in the repository on the screen in the following format. If the repository does not contain any paper, you should display ***---none---.***

```
Paper title, journal name, publication year (for the 1st paper)
Paper title, journal name, publication year (for the 2nd paper)
Paper title, journal name, publication year (for the 3rd paper)
. . .
```

Query a coauthor: You should list all papers that have been written by a coauthor. In this function, the name (first name + last name) of a coauthor is specified as a parameter. The output should include the institution of the coauthor, paper title, and the year, and should be in the following format. Note that if the coauthor have not written any paper (i.e., if the repository does not contain any paper that has been written by the specified author), you should display ***---none---*** after the name of the coauthor.

```
Coauthor name
Institution, paper title, publication year (for the 1st paper)
Institution, paper title, publication year (for the 2nd paper)
```

Institution, paper title, publication year (for the 3rd paper)
. . .

Query the papers which are/were published in a particular journal: You should list all papers published in the journal which is specified as a parameter. The output should include the title of the papers and publication years, and should be in the following format. Note that if there is no paper published in the specified journal (i.e., if the repository does not contain any paper that has been published in the specified journal), you should display ***---none---*** after the journal name.

Journal name
Paper title, publication years (for the 1st paper)
Paper title, publication years (for the 2nd paper)
Paper title, publication years (for the 3rd paper)
. . .

Below is the required public part of the ***PaperRepository*** class that you must write in this assignment. The name of the class must be ***PaperRepository***, and must include these public member functions. We will use these functions to test your code. The interface for the class must be written in a file called ***PaperRepository.h*** and its implementation must be written in a file called ***PaperRepository.cpp***. You can define additional public and private member functions and data members in this class. You can also define additional classes in your solution.

```
class PaperRepository {

public:
        PaperRepository ();
        ~ PaperRepository ();

        void addPaper( const string& paperTitle, const string& journalTitle, const int publicationYear );
        void removePaper( const string& paperTitle );

        void addCoauthor( const string& paperTitle, const string& coauthorFirstName,
                        const string& coauthorLastName, const string& coauthorInstitution );
        void removeCoauthor ( const string& coauthorFirstName,
                        const string& coauthorLastName );

        void showAllPapers();
        void showCoauthor( const string& coauthorFirstName,
                        const string& coauthorLastName );
        void showJournal( const string& journalTitle );

};
```

EXAMPLE TEST CODE:

Below is an example of the test code that can be used for testing your programs. Of course, we will use other test codes in grading. Thus, do not forget to test your program with different test codes as well.

```
#include <iostream>
using namespace std;
#include "PaperRepository.h"
```

```
int main() {
	PaperRepository pr;

	cout << endl;
	pr.showAllPapers();
	cout << endl;

	pr.addPaper( "Smoke Simulation", "SIGGRAPH", 2010 );
	pr.addPaper( "3D Thumbnail", "Algorithms and Technologies", 2011 );
	pr.addPaper( "Dynamic Clustering", "SIGIR", 1999 );
	pr.addPaper( "Efficient SVMs", "IEEE", 1999 );
	pr.addPaper( "Neural Networks", "IEEE", 2002 );
	pr.addPaper( "Neural Networks", "ICML", 2005 );

	cout << endl;
	pr.showAllPapers();
	cout << endl;

	pr.addCoauthor( "Smoke Simulation", "Okan", "Arikan", "Bilkent" );
	pr.addCoauthor( "3D Thumbnail", "Funda", "Atik", "AYESAS" );
	pr.addCoauthor( "3D Thumbnail", "Sena", "Atik", "AYESAS" );
	pr.addCoauthor( "Dynamic Clustering", "Gokhan", "Akcay", "Bilkent" );
	pr.addCoauthor( "Neural Networks", "Aynur", "Dayanik", "Bilkent" );
	pr.addCoauthor( "Neural Networks", "Funda", "Atik", "Bilkent" );
	pr.addCoauthor( "Explosion Simulation", "Aynur", "Dayanik", "Bilkent" );
	pr.addCoauthor( "Neural Networks", "Aynur", "Dayanik", "Bilkent" );
	pr.addCoauthor( "3D Thumbnail", "Yeliz", "Yigit", "Yogurt Tech." );
	pr.addCoauthor( "Efficient SVMs", "Cigdem", "Gunduz Demir", "Bilkent" );
	pr.addCoauthor( "Efficient SVMs", "Funda", "Atik", "Bilkent" );

	cout << endl;
	pr.showCoauthor( "Funda", "Atik" );
	cout << endl;
	pr.showCoauthor( "Aynur", "Dayanik" );
	cout << endl;

	pr.removeCoauthor( "Funda", "Atik" );
	pr.removeCoauthor( "Aynur", "Dayanik" );
	pr.removeCoauthor( "Can", "Koyuncu" );

	cout << endl;
	pr.showCoauthor( "Funda", "Atik" );
	cout << endl;
	pr.showCoauthor( "Aynur", "Dayanik" );
	cout << endl;

	pr.showJournal( "Algorithms and Technologies" );
	cout << endl;
	pr.showJournal( "IEEE" );
	cout << endl;

	pr.removePaper( "Neural Networks" );
	pr.removePaper( "3D Thumbnail" );
	pr.removePaper( "Explosion Simulation" );

	cout << endl;
	pr.showJournal( "Algorithms and Technologies" );
	cout << endl;
	pr.showJournal( "IEEE" );

	cout << endl;
	pr.showAllPapers();
```

```
    cout << endl; return 0;
}
```

OUTPUT OF THE EXAMPLE TEST CODE:

For the test code that is given above, you will see the following output on the screen.

```
---none---

INFO: Paper Smoke Simulation has been added
INFO: Paper 3D Thumbnail has been added
INFO: Paper Dynamic Clustering has been added
INFO: Paper Efficient SVMs has been added
INFO: Paper Neural Networks has been added
ERROR: Paper Neural Networks already exists

Smoke Simulation, SIGGRAPH, 2010
3D Thumbnail, Algorithms and Technologies, 2011
Dynamic Clustering, SIGIR, 1999
Efficient SVMs, IEEE, 1999
Neural Networks, IEEE, 2002

INFO: Coauthor Okan Arikan has been added to Paper Smoke Simulation
INFO: Coauthor Funda Atik has been added to Paper 3D Thumbnail
INFO: Coauthor Sena Atik has been added to Paper 3D Thumbnail
INFO: Coauthor Gokhan Akcay has been added to Paper Dynamic Clustering
INFO: Coauthor Aynur Dayanik has been added to Paper Neural Networks
INFO: Coauthor Funda Atik has been added to Paper Neural Networks
ERROR: Paper Explosion Simulation does not exist
ERROR: Coauthor Aynur Dayanik already is in Paper Neural Networks
INFO: Coauthor Yeliz Yigit has been added to Paper 3D Thumbnail
INFO: Coauthor Cigdem Gunduz Demir has been added to Paper Efficient SVMs
INFO: Coauthor Funda Atik has been added to Paper Efficient SVMs

Funda Atik
Bilkent, Efficient SVMs, 1999
Bilkent, Neural Networks, 2002
AYESAS, 3D Thumbnail, 2011

Aynur Dayanik
Bilkent, Neural Networks, 2002

INFO: Coauthor Funda Atik has been deleted from Paper Efficient SVMs
INFO: Coauthor Funda Atik has been deleted from Paper Neural Networks
INFO: Coauthor Funda Atik has been deleted from Paper 3D Thumbnail
INFO: Coauthor Aynur Dayanik has been deleted from Paper Neural Networks
INFO: Coauthor Can Koyuncu does not have any paper in the repository

Funda Atik
---none---

Aynur Dayanik
---none---

Algorithms and Technologies
3D Thumbnail, 2011

IEEE
Efficient SVMs, 1999
Neural Networks, 2002
```

```
INFO: Paper Neural Networks has been deleted
INFO: Paper 3D Thumbnail has been deleted
ERROR: Paper Explosion Simulation does not exist

Algorithms and Technologies
---none---

IEEE
Efficient SVMs, 1999

Efficient SVMs, IEEE, 1999
Smoke Simulation, SIGGRAPH, 2010
Dynamic Clustering, SIGIR, 1999
```

NOTES ABOUT SUBMISSION:

1. This assignment is due by 23:55 on Friday, July 20, 2018. You should upload your homework to the upload link on Moodle before the deadline. This upload link will be available between July 13 and July 22. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the course syllabus for further discussion of the late homework policy as well as academic integrity.
2. You **MUST** use linked lists in your implementation. Thus, you will define and implement a class for the ADT list and use it in your functions. Note that you are allowed using the C++ codes that are given in your textbook but are not allowed using any other list implementation (e.g., you cannot use the list class defined in another book. Similarly, you are not allowed using any functions in the C++ standard template library (STL).
3. Additionally, you are not allowed using global variables and/or functions, and/or static local variables in your solution. You will get no points if you use global variables and/or functions, and/or static local variables.
4. In this assignment, you must have separate interface and implementation files (i.e., separate .h and .cpp files) for your class. We will test your implementation by writing our own driver .cpp file which will include your header file. For this reason, your class' name MUST BE "***PaperRepository***" and the names of your files MUST BE "***PaperRepository.h***" and "***PaperRepository.cpp***". You should upload these two files (and any additional files if you wrote additional classes in your solution) as a single archive file (.zip file). We also recommend you to write your own driver file to test each of your functions. However, you MUST NOT submit this test code (we will use our own test code). In other words, your submitted code should not include any main function.

   The name of the zip file that you submit must conform to the following name convention: Firstname-Lastname-StudentID.zip.
5. Your code must not have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.
6. You are free to write your programs in any environment (you may use Linux, Mac or Windows). On the other hand, we will test your programs on "`dijkstra.cs.bilkent.edu.tr`" and we will expect your programs to compile and run on the `dijkstra` machine. If we could not get your program properly work on the `dijkstra` machine, you would lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra.cs.bilkent.edu.tr`" before submitting your assignment.

7. This homework will be graded by your TA Ömer Sinan Şahin (sinan.sahin at bilkent edu tr). Thus, you may ask your homework related questions directly to him.