CS 201, Fall 2020
Homework Assignment 3

Due: 23:59, December 14, 2020

In this homework, you will implement a library system by using linked lists. The library system stores information about students and books. The students will be able to check out books from the library and will be able to return them to the library. For each student, the system stores an id, name, and a list of books that she/he checked out. Each book is represented by its id, title, and year. You can assume that there is a single copy of each book in the system. This homework has two parts whose requirements are explained below.

PART A:

This part is a simplified version of the entire system where the user just creates the library system and enters the information about the students. Thus, the system does not contain any books. In this system, you **MUST** store the students in a linked list.

Below is the required `public` part of the `LibrarySystem` class that you must implement in this part.

```
class LibrarySystem {

public:
    LibrarySystem();
    ~LibrarySystem();

    void addStudent(const int studentId, const string studentName);
    void deleteStudent(const int studentId);

    void showStudent(const int studentId) const;
};
```

The details of the implementation are described below.

- The name of the class must be `LibrarySystem` 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 named `SimpleLibrarySystem.h` and its implementation must be written in a file named `SimpleLibrarySystem.cpp`. You must not change anything given in this definition but you can define additional public and private member functions and data members in this class.
- In addition, implement a class named `Student`. This class is quite simple for Part A, but you will have to extend it for Part B. At this point, each student is represented by an id and a name. Do not forget to implement all necessary functions (for example, the constructor and set/get functions) for this class. The interface for this class must be written in a file named `SimpleStudent.h` and its implementation must be written in a file named `SimpleStudent.cpp`. This class must be used in your implementation of the `LibrarySystem` class.
- Add a student: The library system will allow to add a new student with the following information: student id and name. The student ids must be unique positive integers. Therefore, the system should check whether or not the specified student id already exists (i.e., whether or not it is the id of another student), and if the student exists, it should not allow the operation and display a warning message.
- Delete a student: The library system will allow to delete an existing student with student id. If the student does not exist (i.e., if there is no student with the specified id), the system should display a warning message.

- Show detailed information about a particular student: The library system will allow to specify a student id and display detailed information about that particular student. This information includes the student id and student name. If the student does not exist (i.e., if there is no student with the specified student id), the system should display a warning message.
- To test Part A, write your own main function in a separate file. Do not forget to test your code for different cases such that the system is created and the above-mentioned functions are employed. However, do not submit this file. If any of your submitted files contains the main function, you may lose points.

What to submit for Part A?
You should put your `SimpleLibrarySystem.h`, `SimpleLibrarySystem.cpp`, `SimpleStudent.h`, and `SimpleStudent.cpp` files into a folder and zip the folder. In this zip file, there should not be any file containing the main function. The name of this zip file should be: `PartA_secX_Firstname_Lastname_StudentID.zip` where `X` is your section number. Then, follow the steps that will be explained at the end of this document for the submission of Part A.

What to be careful about implementation and submission for Part A?
You have to read the "notes about implementation" and "notes about submission" parts that are given at the end of this document.

PART B:

In this part, you will extend the simple library system. In this version, the system **MUST** use a linked list to store the students, and for each student, a linked list to store the books checked out by this student. There **MUST** be an additional linked list that stores the books that are not checked out by any student. All lists can be unsorted.

Below is the required `public` part of the `LibrarySystem` class that you must implement in this part.

```
class LibrarySystem {

public:
    LibrarySystem();
    ~LibrarySystem();

    void addBook(const int bookId, const string bookName, const int bookYear);
    void deleteBook(const int bookId);

    void addStudent(const int studentId, const string studentName);
    void deleteStudent(const int studentId);

    void checkoutBook(const int bookId, const int studentId);
    void returnBook(const int bookId);

    void showAllBooks() const;
    void showBook(const int bookId) const;
    void showStudent(const int studentId) const;
};
```

The details of the implementation are described below.

- The name of the class must be `LibrarySystem` 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 named `LibrarySystem.h` and its implementation must be written in a file named `LibrarySystem.cpp`. You must not change anything given in this definition but you can define additional public and private member functions and data members in this class.

- In addition, implement a class named `Book`. Each book is represented by its id, title, and year. Do not forget to implement all necessary functions (for example, the constructor and set/get functions) for this class. The interface for this class must be written in a file named `Book.h` and its implementation must be written in a file named `Book.cpp`. This class must be used in your implementation of the `LibrarySystem` class.
- Furthermore, implement an extended version of the class named `Student` that supports the functionality required in Part B. Do not forget to implement all necessary functions (for example, the constructor and set/get functions) for this class. The interface for this class must be written in a file named `Student.h` and its implementation must be written in a file named `Student.cpp`. This class must be used in your implementation of the `LibrarySystem` class.
- Add a book: The library system will allow to add a new book to its collection with the following information: book id, book title, and the year it was published. The book ids must be unique positive integers. Hence, the system should check whether or not the specified book id already exists (i.e., whether or not it is the id of another book), and if the book exists, it should not allow the operation and display a warning message.
- Delete a book: The library system will allow to delete an existing book specified with its book id. If the book does not exist (i.e., if there is no book with the specified id), the system should display a warning message. Note that it should be possible to delete books which are checked out. In that case, the book should be returned back to the library system before deletion (see the sample code output).
- Add a student: The library system will allow to add a new student with the following information: student id and name. The student ids must be unique positive integers. Therefore, the system should check whether or not the specified student id already exists (i.e., whether or not it is the id of another student), and if the student exists, it should not allow the operation and display a warning message.
- Delete a student: The library system will allow to delete an existing student with student id. If the student does not exist (i.e., if there is no student with the specified id), the system should display a warning message. If the student has checked out books, those books must be returned to the library system as part of the delete operation for the student (see the sample code output).
- Checkout a book by a student: The library system will allow to check out a particular book by a particular student. For that, the book id and the student id have to be specified. The system should first check whether or not this book exists; if it does not, it should not allow the operation and display a warning message. The system should also check whether or not this student exists; if she/he does not, it should not allow the operation and display a warning message. If both the book and the student exist and the book is not already checked out, the check out operation must be performed. Note that a book can be checked out by only one student. However, a student can check out multiple books.
- Return a book: The library system will allow a student to return a book. For that, the book id has to be specified. If the book does not exist, the system will display a warning message. It will also check whether or not this book is checked out; if not, a warning message will be displayed, otherwise, the operation will be carried out.
- Show the list of all books: The library system will allow to display the list of all the books. This list includes, for each book, the book id, book title, and publication year. For each book that is checked out, id and name of the student who checked out the book should also be displayed.
- Show detailed information about a particular book: The library system will allow to specify a book id and display detailed information about that particular book. This information includes the book id, book title, publication year and whether or not the book was checked out. If the book is checked out, id and name of the student who checked out the book should be displayed. If the book does not exist (i.e., if there is no book with the specified book id), the system should display a warning message.

- Show detailed information about a particular student: The library system will allow to specify a student id and display detailed information about that particular student. This information includes the student id, student name, and the list of books checked out by this student including the book id, book title, and publication year. If the student does not exist (i.e., if there is no student with the specified student id), the system should display a warning message.

Here is an example test program that uses this class and the corresponding output. We will use a similar program to test your solution so make sure that the name of the class is `LibrarySystem`, its interface is in the file named `LibrarySystem.h`, and the required functions are defined as shown above. You can assume that the book ids and student ids are entered as positive integers.

Example test code:

```
#include <iostream>
using namespace std;

#include "LibrarySystem.h"

int main() {

    LibrarySystem LS;

    LS.addBook(1000, "Machine Learning", 1997);
    LS.addBook(1200, "Data Mining", 1991);
    LS.addBook(1300, "Problem S. with C++", 1991);
    LS.addBook(1400, "C++ How to Program", 2005);
    LS.addBook(1200, "Data Mining", 1991);
    LS.deleteBook(1300);
    LS.deleteBook(2000);
    LS.addBook(1500, "Pattern Recognition", 2000);
    cout << endl;
    LS.addStudent(21000000, "Selim Aksoy");
    LS.addStudent(21000011, "Ercument Cicek");
    LS.addStudent(21000011, "Aydamir Mirzayev");
    LS.addStudent(21000020, "Mert Duman");
    LS.addStudent(21000001, "Han Solo");
    LS.addStudent(21000005, "Jack Ryan");
    LS.deleteStudent(21000011);
    LS.deleteStudent(21000050);
    cout << endl;
    LS.checkoutBook(1200, 21000000);
    LS.checkoutBook(1400, 21000020);
    LS.checkoutBook(2050, 21000011);
    LS.checkoutBook(1000, 21000444);
    LS.checkoutBook(1500, 21000000);
    LS.checkoutBook(1400, 21000001);
    cout << endl;
    LS.showStudent(21000000);
    cout << endl;
    LS.showStudent(21000005);
    cout << endl;
    LS.showStudent(21000011);
    cout << endl;
    LS.showBook(1000);
    LS.showBook(1200);
    cout << endl;
    LS.showAllBooks();
    cout << endl;
    LS.returnBook(1200);
    LS.returnBook(1000);
    cout << endl;
    LS.checkoutBook(1200, 21000020);
    LS.checkoutBook(1000, 21000000);
    cout << endl;
    LS.showAllBooks();
    cout << endl;
    LS.deleteStudent(21000020);
    cout << endl;
    LS.deleteBook(1000);
    cout << endl;
    LS.showStudent(21000000);
    cout << endl;
    LS.showAllBooks();

    return 0;
}
```

Output of the example test code:

```
Book 1000 has been added
Book 1200 has been added
Book 1300 has been added
Book 1400 has been added
Book 1200 already exists
Book 1300 has not been checked out
Book 1300 has been deleted
Book 2000 does not exist
Book 1500 has been added

Student 21000000 has been added
Student 21000011 has been added
Student 21000011 already exists
Student 21000020 has been added
Student 21000001 has been added
Student 21000005 has been added
Student 21000011 has been deleted
Student 21000050 does not exist

Book 1200 has been checked out by student 21000000
Book 1400 has been checked out by student 21000020
Book 2050 does not exist for checkout
Student 21000444 does not exist for checkout
Book 1500 has been checked out by student 21000000
Book 1400 has been already checked out by another student

Student id: 21000000 student name: Selim Aksoy
Student 21000000 has checked out the following books:
Book id   Book name              Year
1200      Data Mining            1991
1500      Pattern Recognition    2000
```

```
Student id: 21000005 student name: Jack Ryan
Student 21000005 has no books

Student 21000011 does not exist

1000      Machine Learning       1997      Not checked out
1200      Data Mining            1991      Checked out by student 21000000

Book id   Book name              Year      Status
1000      Machine Learning       1997      Not checked out
1200      Data Mining            1991      Checked out by student 21000000
1400      C++ How to Program     2005      Checked out by student 21000020
1500      Pattern Recognition    2000      Checked out by student 21000000

Book 1200 has been returned
Book 1000 has not been checked out

Book 1200 has been checked out by student 21000020
Book 1000 has been checked out by student 21000000

Book id   Book name              Year      Status
1000      Machine Learning       1997      Checked out by student 21000000
1200      Data Mining            1991      Checked out by student 21000020
1400      C++ How to Program     2005      Checked out by student 21000020
1500      Pattern Recognition    2000      Checked out by student 21000000

Book 1400 has been returned
Book 1200 has been returned
Student 21000020 has been deleted

Book 1000 has been returned
Book 1000 has been deleted

Student id: 21000000 student name: Selim Aksoy
Student 21000000 has checked out the following books:
Book id   Book name              Year
1500      Pattern Recognition    2000

Book id   Book name              Year      Status
1200      Data Mining            1991      Not checked out
1400      C++ How to Program     2005      Not checked out
1500      Pattern Recognition    2000      Checked out by student 21000000
```

What to submit for Part B?

You should put your `LibrarySystem.h`, `LibrarySystem.cpp`, `Student.h`, `Student.cpp` `Book.h`, and `Book.cpp` files into a folder and zip the folder. In this zip file, there should not be any file containing the main function. The name of this zip file should be: `PartB_secX_Firstname_Lastname_StudentID.zip` where `X` is your section number. Then, follow the steps that will be explained at the end of this document for the submission of Part B.

What to be careful about implementation and submission for Part B?

You have to read the "notes about implementation" and "notes about submission" parts that are given at the end of this document.

IMPORTANT NOTES:

Do not start your homework before reading these notes!!!

Output message for each operation should match the format shown in the output of the example code.

NOTES ABOUT IMPLEMENTATION (for both Part A and Part B):

1. You MUST use linked lists in your implementation according to the description provided for each part. You will get no points if you use automatically allocated arrays, dynamic arrays, or any other data structure such as vector/array from the standard library.
2. You MUST use your own implementation of linked lists. In other words, you cannot use any existing linked list code from any other source. However, you can adapt the linked list codes in the Carrano book or in our lecture slides. You will get no points if you do not use linked lists as indicated in the highlighted sections.
3. You are NOT ALLOWED to modify the given parts of the class definitions. Note that you can define additional public and private member functions and data members in the given classes.
4. You are NOT ALLOWED to use any global variables or any global functions.
5. Your code MUST NOT have any memory leaks for Part A and Part B. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.
6. Make sure that each file that you submit (each and every file in the zip archive) contains your name, section, and student number at the top as comments.

NOTES ABOUT SUBMISSION (for both Part A and Part B):

1. This assignment is due by 23:59 on December 14, 2020. This homework will be graded by your TA Aydamir Mirzayev (aydamir.mirzayev at bilkent edu tr). Please direct your homework related questions to him.
2. Conform to the rules given separately for Part A and Part B. That is, the names of the classes, the names of the .h and .cpp files, and the names of the zip files should conform to the specifications separately given for Part A and Part B. Otherwise, you may lose a considerable amount of points.
3. You need to upload two zip files (one for Part A and the other for Part B) using the upload link on Moodle (all sections) by the deadline. Read "what to submit for Part A" and "what to submit for Part B" sections very carefully.
4. Do not submit any files containing the main function. We will write our own main function to test your implementations.
5. No hard copy submission is needed. The standard rules about late homework submissions apply. Please see the course web page for further discussion of the late homework policy as well as academic integrity.
6. You are free to write your programs in any environment (you may use either Linux or Windows). Yet, we will test your programs on "dijkstra.ug.bcc.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.ug.bcc.bilkent.edu.tr before submitting your assignment.