CS 3353 Data Structure and Algorithm Analysis I
Assignment 02: Full Marks 100
Due Date: 10/02/2022, 11:59 PM CT **Due Date: 10/09/2022, 11:59 PM CT**

Read the following instructions before you proceed with the assignment:

- The assignment needs to be completed individually in Java.
- Make sure the program runs in CSX machine.
- Please go through the grading rubrics and submission instructions carefully to avoid losing points.

In this assignment, you will implement two data structures: *doubly linked list* and *singly linked list*. You need to use your own list ADT and not use list from Collection frameworks.

Input File Description:

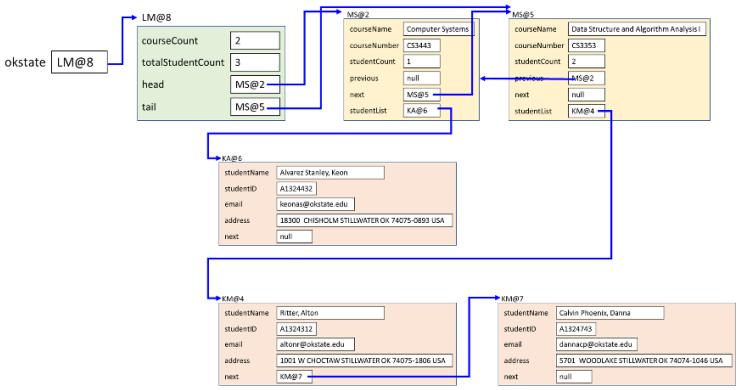
In this assignment, you will need to use the provided input file: *inputFile.txt*. The file consists of the following columns:

1. *Course Number*: a 4-digit course number
2. *Course Name*: Name of the course
3. *Student's Name*: name of the student given in <middle_name last_name>, <first_name> or <last_name>, <first_name>
4. *Student's ID*: Unique 7-character ID number of the student
5. *Email*: Unique email address of the student
6. *Address*: Emergency contact address of the student

Disclaimer: All data appearing in this text file are fictitious and crated randomly using Excel. Any resemblance to any actual student is purely coincidental.

Programming Requirement:

You need to write a program in Java that reads the input file and organize the data using two data structures: *doubly linked list* and *singly linked list*. *Doubly linked list* will be used to organize course information like course number, course name, number of students enrolled in the course and the pointer to point to the *singly linked list*, which contains the information about the students (student's name, student's ID, email and address) enrolled in that course. The courses in the *doubly linked list* are organized in an alphabetical order by course number. The student's records are arranged in the *singly linked list* in an alphabetical order by the first name. Your implementation should look as follows:



In the above diagram, the object LM@8 is called the header. It contains four values:

1. courseCount: specifies the number of courses in the *doubly linked list*. In the given example, the value of *courseCount* is 2 as there are two courses: CS3443: Computer Systems and CS3353: Data Structure and Algorithm Analysis I.
2. totalStudentCount: specifies the total number of students enrolled in all the courses. In the given example, the value of *totalStudentCount* is 3 as there is one student enrolled in CS3443 and two students enrolled in CS3353.
3. head: a pointer to the first node in the *doubly linked list*. In the given example, the first node in the *doubly linked list* is MS@2.
4. tail: a pointer to the last node in the *doubly linked list*. In the given example, the last node in the *doubly linked list* is MS@5.

In the above example, there are two nodes in the doubly linked list, shown by MS@2 and MS@5. The nodes in the doubly linked list contains six values:

1. courseName: specifies the course name, which is obtained from the input file *inputFile.txt*.
2. courseNumber: specifies the corresponding course number, which is obtained from the input file *inputFile.txt*.
3. studentCount: number of students enrolled in the corresponding course. In the given example, the value of *StudentCount* for the node MS@2 is 1 as there is one student enrolled in CS3443.
4. previous: a pointer to the previous node in the *doubly linked list*.
5. next: a pointer to the next node in the *doubly linked list*.
6. studentList: a pointer to the first node in the *singly linked list*. In the given example, the value of *studentList* for the node MS@2 is KA@6 as that is the first node in the singly linked list.

In the above example, there are two singly linked lists: one for the course CS3443 and another for the course CS3353. For the course CS3443, there is a single node shown by KA@6 while for the course CS3353, there are two nodes, shown by KM@4 and KM@7. Each node in the singly linked list contains five values:

1. studentName: specifies the name of the student enrolled in the corresponding course. The list of students enrolled in each course is obtained from the input file *inputFile.txt*.
2. studentID: specifies the student ID of the corresponding student, which is obtained from the input file *inputFile.txt*.
3. email: email address of the corresponding students, which is obtained from the input file *inputFile.txt*.
4. address: emergency contact address of the student, which is obtained from the input file *inputFile.txt*.
5. next: a pointer to the next node in the *singly linked list*. In the given example, the value of *next* is null for the node KA@6, as this is the only student enrolled in the course CS3443. For the course CS3353, the value of *next* in KM@4 is KM@7 as the student named Alton Ritter comes alphabetically before the student named Danna Calvin Phoenix.

In the given diagram, to get the first course name in the *doubly linked list*, one can traverse the list as `okstate.head.courseName`. Similarly, to get the second student's name in the course CS3353, one can traverse the list as `okstate.head.next.studentList.next.studentName`.

Tasks:

You need to write a menu driven program with the following options:

1. Read the input data
2. Delete a course
3. Insert a new course
4. Delete a student
5. Insert a new student
6. Transfer a student from one course to another
7. Display the course list
8. Display the student list
9. Exit

When the user chooses option (1), the program should read the given input file and organize in the appropriate data structures as discussion in the section **Programming Requirement**:. Once the data is read, the program should display the summary information from the header on the screen. The output in this option should be as shown below:

```
Input file is read successfully..
Summary of the record:
Number of courses registered: 2
Number of total students: 3
```

Your program should not allow user to choose option (1) multiple times. Also, only after choosing option (1), user can pick other option(s) from 2 to 8. However, user can choose option (9) without reading the file.

When the user chooses option (2), the program should ask user which course to delete. Once the course is deleted, the program should display the summary information from the header on the screen. The output on this option, for the given example, should be as follow:

```
Enter the course number to delete: CS3443
Summary of the record:
Number of courses registered: 1
Number of total students: 2
```

When the user chooses option (3), the program should ask user which new course to insert. Once the new course is added, the program should display the summary information from the header on the screen. The output on this option, for the given example, should be as follow:

```
Enter the new course number to add: CS4323
Enter the new course name for CS4323: Design and Implementation of Operating Systems I
Summary of the record:
Number of courses registered: 3
Number of total students: 2
```

When the user chooses option (4), the program should ask which student from which course is to be deleted. Once the student is deleted from that course, the program should display the summary

information from the header on the screen. The output on this option, for the given example, should be as follow:

```
Enter the student ID number to delete: A1324312
Enter the course number from which the student is to be dropped from: CS3353
Summary of the record:
Number of courses registered: 2
Number of total students: 2
```

When the user chooses option (5), the program should ask user for which course new student to be inserted and then the student's information. Once the new student is added for the course, the program should display the summary information from the header on the screen. The output on this option, for the given example, should be as follow:

```
Enter the course number the student wants to enroll to: CS3443
Enter the student's name: Seth River Vickery
Enter the student's ID: A205820
Enter the student's emergency contact address: 7874 Wellington Ave.Oklahoma City, OK 73169
Summary of the record:
Number of courses registered: 2
Number of total students: 4
```

When the user chooses option (6), the program should ask user for which course, which student wants to be transferred to. Once the student has been transferred from one course to another, the program should display the summary information from the header on the screen. The output on this option, for the given example, should be as follow:

```
Enter the student's name: Alton Ritter
Enter the course number the student wants to drop from: CS3353
Enter the course number the student wants to enroll to: CS3443
Summary of the record:
Number of courses registered: 2
Number of total students: 3
```

When the user chooses option (7), the program should display the list of all the courses along with its summary. The output on this option, for the given example, should be as follow:

```
The list of courses registered are as follows:
Course Number: CS3443
Course Name: Computer Systems
Number of students enrolled: 1

Course Number: CS3353
Course Name: Data Structure and Algorithm Analysis I
Number of students enrolled: 2
```

When the user chooses option (8), the program should display the list of all the students enrolled in that course. The output on this option, for the given example, should be as follow:

Enter the course number: CS3353

The list of students enrolled in the course CS3353 are as follows:

Student's ID	Students Name	Email	Address
A1324312	Alton Ritter	altonr@okstate.edu	1001 W CHOCTAW STILLWATER OK 74075-1806 USA
1324743	Danna Calvin Phoenix	dannacp@okstate.edu	5701 WOODLAKE STILLWATER OK 74074-1046 USA

Please make a note that the outputs are properly formatted in as shown in the example. All the user input(s) have been shown in the red.

User can pick option (9) at any time if he/she chooses to exit the program. Only on picking option (9), the program should terminate. If the user chooses any other option, then the user should be displayed the menu option again, after the selected task is completed.

Please make a note of the advantages of the *singly-linked list*:

- List can be of any size.
- Insertion or removal of a value at the beginning can be done at a constant time.
- Referencing an element *i* on the list takes a time proportional to *i*, as one has to sequence through all the nodes from *0* to *i – 1* to find it.

Also, make a note of the advantages of the doubly-linked list:

- It takes a constant time to append (insert an element at the end of the list) a value to the list or prepend a value (insert an element at the beginning of the list).
- It takes a constant time to insert a value or delete a value before or after a given element in the list.

It will be your job to implement all these tasks in the most efficient way. For this, you need to compute the worst case time complexity of your algorithms.

Grading Rubric:

Your assignment will be graded based on the following grading Rubrics:

• Design of the overall data structure as discussed in the Programming Requirement:		**[30 Points]**
• Successful completion of the tasks and the display of appropriate results		**[45 Points]**
○ Option (1): Read the input data	[5 Points]	
○ Option (2): Delete a course	[5 Points]	
○ Option (3): Insert a new course	[5 Points]	
○ Option (4): Delete a student	[5 Points]	
○ Option (5): Insert a new student	[5 Points]	
○ Option (6): Transfer a student from one course to another	[5 Points]	
○ Option (7): Display the course list	[5 Points]	
○ Option (8): Display the student list	[5 Points]	
○ Option (9): Exit	[5 Points]	
• Worst case complexity analysis		**[25 Points]**
○ Both deletion of courses and students	[10 Points]	
○ Both insertion of a new course and new students	[10 Points]	
○ Transfer of student from one course to another.	[5 Points]	

Tips to avoid any point deduction:

- Failure to follow standard programming practices will lead to points deduction, even if the program is running correctly, like
  - No comments on code
  - Not writing meaningful identifiers
- If the program does not compile successfully on the CSX machine, then 20 points will be automatically deducted.

Submission Guidelines:

- All the files should be submitted through Canvas. Assignment submitted through email will not be accepted. CSX machine is only used to run the program and not for assignment submission.

- Your assignment will be checked in the CSX machine. You need to include readMe.txt file that shows:
  - how to run your program.
  - which csx server has been tested on
  - any assumption you made on the assignment.

- The worst-case analysis needs to be performed in a separate pdf file. It needs to be computer typed and not handwritten. Name this file as:

  assignment01_lastName_firstName_worstCaseComplexity.pdf

- All the java files need to have following format:

  assignment01_lastName_firstName_<appropriate_file_name>.pdf

  where, use meaningful file name in the place of appropriate_file_name.

- In addition to .java file, all the codes should be submitted in .pdf format as well. Please copy the codes in the text form. Do not screenshot or include any image of the code in the pdf. This is used to check for plagiarism.

- The main driver file needs to have the following header information:
    - Author Name:
    - Email: <Use only official email address>
    - Date:
    - Program Description:

- Use comments throughout your program.
    - Each class and the method should be properly commented:
        - description of arguments and return data
        - description of method