King Abdulaziz University Faculty of Computing and Information Technology Computer Science Department

**CPCS204, Spring 2023 Program 1**

**Course Coordination System**

**Assigned: Thursday March 30, 2023 Due: Thursday April 13, 2022**

**Purpose**

1. Learn to implement a linked list for a real-world problem.
2. Review file I/O (input/output).

**Read Carefully:**

This program is worth 5% of your final grade.

**WARNING:** This is an individual project; you must solve it by yourself. Any form of cheating will result in receiving **zero** in the assignment.

The deadline for this project is **Thursday April 13, 2023 by 11:59 PM.**

**LATE SUBMISSION:** **No assignment will be accepted after the deadline**

**Blackboard Submission:**

This project must be submitted online via Blackboard.

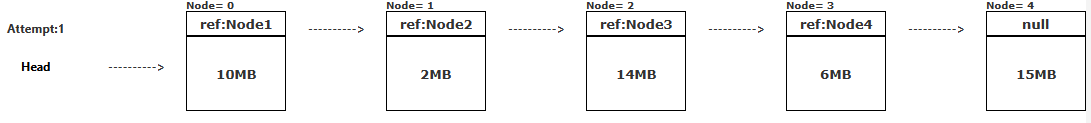
The source file(s) of your program should be zipped up. You must name the zip file using the following naming convention: SectionNumber\_StudentID\_ProgramNumber.zip

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Question:** | 1 | 2 | 3 | **Total** |
| **Points:** | 20 | 20 | 60 | **100** |

*Question 1:* (20 points)

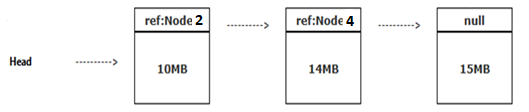
1. Suppose a singly linked list contains several nodes; each node has an object. Describe steps to delete the nodes which are smaller than the average of all nodes. Your discretion of steps must work with any number of nodes. If you want, you can write the steps using Java syntax.

**Note:** The size of each node can be obtained by the **getSize()** method, and the **head** is a reference to the head node of  the singly linked list



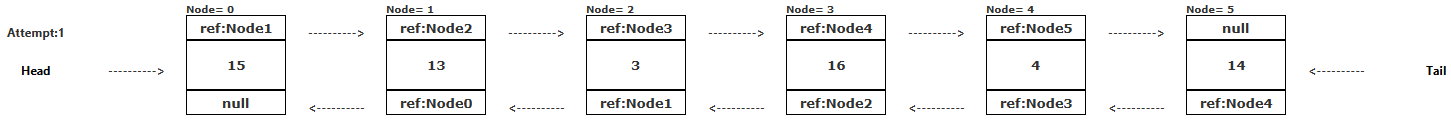
------------------------------------------------------------------------------------------------

The following figure shows how the singly linked list looks after deletion.



*Question 2:*(20 points)

1. Suppose that a doubly linked list has been created, and each node contains a numeric number as follows:



**Write a method** called ***sum()***  that takes in the head and tail pointers of the created doubly linked list. This method will print the sum of the first and last nodes, the sum of the second and second-to-last, and so on.

The method must meet the following requirements:

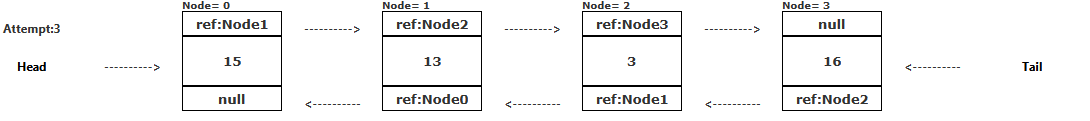
1. Work with any number of nodes.
2. Navigate the doubly linked list in both directions to efficiently handle the increasing number of nodes.
3. If the doubly linked list has an odd number of nodes, then the value of the middle node should be displayed as it is.

**Note:** The data of each node can be obtained by the ***data*** property (variable).

------------------------------------------------------------------------------------------------

Consider the following expected output for two doubly linked lists:

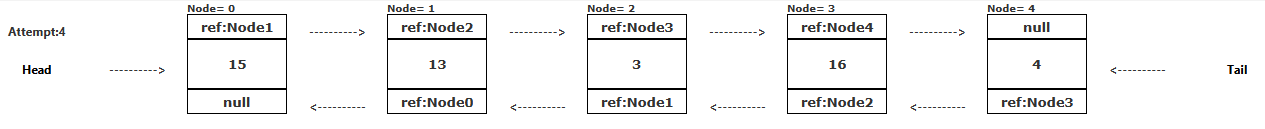
Example 1:



**Output1:**

31 16

Example 2:



**Output2:**

19 29 3

**Question 3: (60 points)**

**Program 1: Course Coordination System**

**Objective:**

The goal of this program is to develop a course coordination system that aims at assigning courses to instructors. The primary objective of this program is to implement a linked list. The secondary objective is to practice with File I/O.

**Program Description:**

Write a program to assign instructors to specific courses. Each course requires a coordinator. Note that the coordinator can be an instructor as well. You are required to write the following methods:

1. Write a function to add an instructor to a specified course (**at the end of the linked list**).
2. Write a function to search for an instructor based on his/her ID.
3. Write a function to delete an instructor from the linked list based on his/her ID.
4. Write a function to swap between two instructors.

The program deals with three files. Two input files and one output file. Information on the same line is separated by spaces. The description of these files as follow:

- The first input file (**intialInformation**.**txt**) contains the important information for the system which includes information about the instructors, the courses, and the coordinators. The information in this file is arranged as follow:

* The first line contains **number of instructors**, **number of classes** and **number of coordinators**.
* The second line contains **the courses numbers**.
* The third line contains **the coordinators IDs**.
* The following lines (from line 4 to 18) contain the instructor’s information, that are, **instructor ID**, **first name**, and **last name**.
* The lines from 19 to 21 contain the information of the courses including **course number** and **course name**.
* The commands for the system are found in the second file called **commands.txt**. The commands in this file as follow:
* **STARTUP**: This command will use the first input file (intialInformation.txt) to initialize the system by creating 3 linked lists equal to the number of courses specified in the file. Every linked list holds the instructors’ information of one course. Each created linked list will have 5 instructors in order as they appear in the file. Then it assigns one coordinator for each course. For clarification, the first linked list is for CPCS204 course, the coordinator is 001 and the instructors are the first 5 instructors as they sequentially appear in intialInformation.txt. The second linked list is for CPCS203, the coordinator is 006, and the instructors are the second 5 instructors as they sequentially appear in intialInformation.txt. The third linked list follows the same process.
* **DISPLAY\_ALL\_COURSES:** This command will display all instructors in all courses as it is shown in the **ouput.txt** file.
* **DISPLAY\_COURSE\_FOR\_** **INSTRUCTOR:** This command requires one input which determines the instructor ID. It will display information about the instructor, the assigned course and the coordinator of this course. If no instructor is found, the system shows “No instructor of this ID is found”.
* **NUM\_OF\_ INSTRUCTORS:** This command requires one input which determines the course number. It will display the number of instructors in the specified course.
* **DELETE\_INSTRUCTOR**: This command requires one value that determines the instructor ID. It will delete the instructor from the course. If the deleted instructor is a coordinator, the system refuses the deletion by showing “You cannot delete a coordinator”.
* **SWAP\_ INSTRUCTOR**: This command requires two inputs which determine two instructors IDs. It will swap the two instructors. Note that you cannot swap instructors with a coordinator unless the two of them are coordinators. If so, the system shows “ Coordinator - name of the coordinator - cannot be swap with the instructor - name of the instructor ” as shown in the output file.
* **QUIT**: This command will stop the program.
* The output of the program should be written to the file name output.txt, which content should be similar to the contents of the file provided to you.

**Implementation**

For this program, you will create the following classes:

• Instructors.java: This class will be used to create objects of type instructor. Each instructor object will store the instructor ID, instructor first name, instructor last name, course number, and coordinator.

• Course.java: All the methods will be implemented in this class.

• MainProgram.java: This is the class that will contain the main.

**Sample Input & Output File**

We have provided you with a sample for two input files and one output file.

**Grading Details**

Your program will be graded upon the following criteria:

1. Adhering to the implementation specifications listed on this write-up.
2. Your algorithmic design.
3. Correctness.
4. **Use of the three classes, as specified. If your program is missing these elements, you will lose marks.**
5. The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
6. Compatibility to the **newest version** of NetBeans. (If your program does not compile in NetBeans, you will get a large deduction from your grade.)
7. Your program should include a header comment with the following information: your name, **email**, account number, section number, assignment title, and date.
8. Your output MUST adhere to the EXACT output format shown in the sample output file.

**Deliverables**

You should submit a zip file with four files inside:

1. *ConceptPart.doc (Algorithm and Method Write up)*
2. *Instructors.java*
3. *Course.java*
4. *MainProgram.java*

\*\*\*These three files should all be INSIDE the same package called **InstructorAssignment**. If they are not in this specific package, you will lose points.

**NOTE: your name, ID, section number AND EMAIL should be included as comments in all files!**

**UML Diagrams:**

For this program, you will create **three** Classes (UML diagram shown below):

|  |  |  |
| --- | --- | --- |
| **Instructor** |  | **Course** |
| *Data Members*  private int ID;  private String Fname;  private String lName;  private String Course;  private Boolean Coordinator;  private Instructor next; |  | *Data Members*  private String courseName;  private int courseNumber;  private Instructor head; |
| *Operations/Methods*  Instructor () // one or more Constructors      And any other methods you need. |  | *Operations/Methods*  Course()  addInstructor()  searchInstructorByID()  swapInstructor ()  deleteInstructorByID()  ALL necessary methods for linked-list operations |
|  |  |  |
| *Data Members*  As needed |  |
| *Operations/Methods*  public static void main()  ALL necessary methods for all operations |  |