SCS214: Data Structures

Assignment-2: Linked lists

Instructions

1. The assignment is submitted in groups of **maximum 3** students from the same **lab OR** same **TA.**
2. Deadline of submission is **27/3/2024**
3. The submission will be on Google classroom.
4. Your submission should include a single **cpp file**, named LabGroup_ID1_ID2_ID3 (ex.: S13_20220022_20220023_20220024.cpp).
5. No late submission is allowed.
6. No submission through e-mails.
7. No rar files
8. No exe file submission.
9. **In case of Cheating, you will get a negative grade whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason.**
10. You must write clean code and follow a good coding style including choosing meaningful variable names.
11. In case of wrong submission, wrong file extension/type, missing files, plagiarism, extra submitted files, wrong naming, the assignment will not be accepted and no correction for these mistakes is allowed, and you will lose your grade.

Task# 1 Sorted linked list

You have a class called “Student” that contains 3 attributes: Name, ID, GPA.

Then Create a Singly linked list of class Student Called “ StudentList ”

Following are the operations supported by a list:

- StudentList () – default constructor.
- ~ StudentList () – a destructor.
- Insertion – Adds a student to the list. While keeping the list sorted by ID
- Delete – Deletes an element with the given ID
- Display – Displays the complete list.

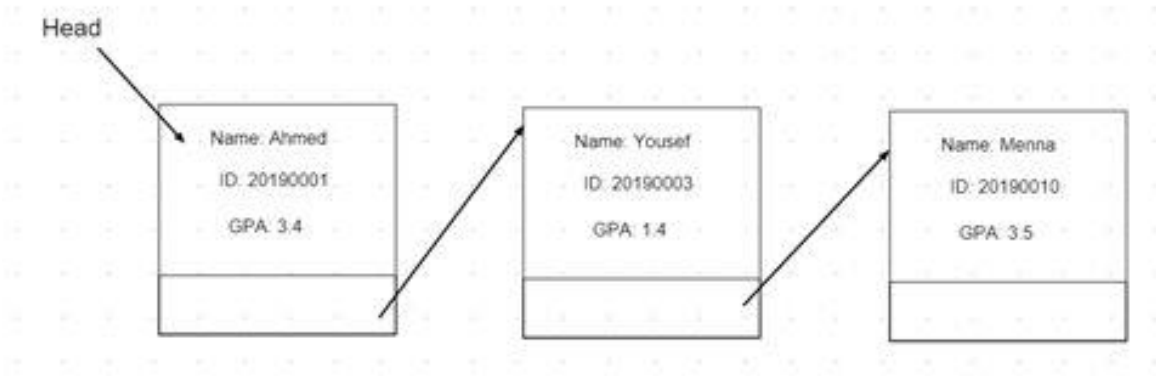
- Search – Searches an element using the given key. And prints it.
- int size() – returns the current number of students in the list.

Example:

After executing the following code, your list should look like:

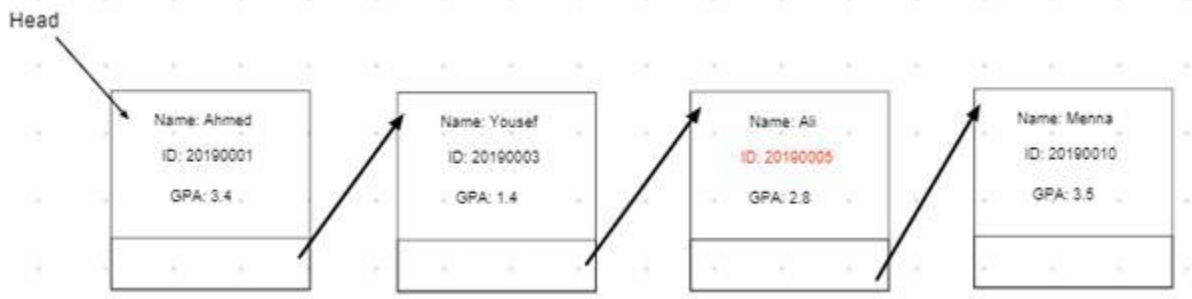
```
StudentList list;
Student s1("Ahmed", 20190001, 3.4);
Student s2("Menna ", 20190010, 3.5);
Student s3("Yousef ", 20190003, 1.4);

list.insert(s1);
list.insert(s2);
list.insert(s3);
```



Inserting a new Student should keep the list ordered by ID:

```
Student s4("Ali ", 20190005, 2.5);
list.insert(s4);
```



SCS214: Data Structures

Assignment-2: Linked lists

Task 2: String Manipulation using Doubly Linked List

You are asked to implement string manipulation system using a doubly linked list. The system should provide various functionalities for manipulating strings, such as concatenation, insertion, deletion, substring extraction, and searching.

Implement the following functionalities:

- Initialization: Create a doubly linked list to store characters of the string.
- Insertion: Implement a function to insert a character at a specified position in the string.
- Deletion: Implement a function to delete a character at a specified position in the string.
- Concatenation: Implement a function to concatenate two strings represented by doubly linked lists. Example: L1.concat(L2). After the call the concatenated string will be in L1, while L2 won't change.
- Substring Extraction: Implement a function to extract a substring from the string, given the starting index and length.
- Search: Implement a function to search for a given substring within the string and return its starting index (1^{st} occurrence).
- Replacement: Implement a function to replace occurrences of a substring with another substring within the string.
- Reverse: Implement a function to reverse the string.

The output should be like that:

SCS214: Data Structures

Assignment-2: Linked lists

```
Enter string to add to list 1: Hello
Enter string to add to list 2: World
Concatenated Lists: HelloWorld
Choose a character by index to remove: 1
List after removal: HlloWorld
Enter index and length to get substring: 6 3
Substring: rld
Search for a string in the list: World
Found at index 4
Enter 2 substrings to replace one with another: Wor NEWPART
List after replacement: HlloNEWPARTld
Reversed List: dlTRAPWENollH
```

Grading Criteria:

Task 1:

Main	**5**
Constructor and Destructor	**5**
Insertion method	**10**
Deletion Method	**10**
Display method	**7**
Search method	**8**
Size method	**5**
Total	**50**

Task 2:

Main	**5**
Insertion	**5**
Deletion	**5**
Concatenation	**10**
Substring Extraction	**5**
Search	**5**
Replacement	**7.5**
Reverse	**7.5**
Total	**50**