CIS350 Winter 2025: Assignment 1

- **Implementing/Extending a Doubly Linked List &**
- **Implementing/Extending a Score Management System**

Objective:

In this assignment, you will extend the implementation of a Doubly Linked List (DLL) and implement additional functions for a score management system using Arrays, Vectors, or Lists.

You will demonstrate your understanding of linked data structures, dynamic arrays, and algorithm design by implementing, testing, and analyzing the behavior of your extended Doubly Linked List and the Score Management System. Choose only 1 programming language from:

C++: See section 3.1 for Score Management System and section 3.3 for Doubly Linked Lists

Java: See section 3.1 for Score Management System and section 3.4 for Doubly Linked Lists

Python: See section 5.5 for Score Management System and section 7.3 for Doubly Linked Lists

Part 1: Doubly Linked List

1. **Starting Code:**
    - You are provided with a basic implementation of a Doubly Linked List (DLinkedList), which includes basic functions such as adding and removing elements from the front and back and printing the list. Study the provided code carefully to understand its structure and functionality. **Note that the given addBefore() method adds an element before the given node. You should be able to demonstrate in testing how an element can be added after an existing element other than the front and back of the list.**

      Note: Your submission will contain same name and number of files as given. Only provided code content will change.
    - The starting code will differ slightly based on the language you choose. Modify as appropriate:
        - C++: Header files and implementation files.
        - Java: Classes with constructors and methods.
        - Python: Class definitions with methods.
2. **Tasks:**

    You need to implement the following additional functions in the DLinkedList class:

    1. **Implement the Find Function:**
        - Function Signature:
            a. C++: int find(const Ele& element);
            b. Java: int find(String element);
            c. Python: def find(self, element):
        - **Description:** Find the first occurrence of the specified element (element) in the list. Return an integer indicating the position of the element found in the list containing the element if found, otherwise return -1. If the element is found, print a message indicating its position in the list.

2. **Insert After a Given Element:**
    - Function Signature:
        a. C++: void insertAfter(const Ele& existingElement, const Ele& newElement);
        b. Java: void insertAfter(String existingElement, String newElement);
        c. Python: def insert_after(self, existing_element, new_element):
    - **Description:** Insert a new element (newElement) after a given element (existingElement). If the existingElement is not found, print a suitable message.
3. **Reverse the List:**
    - Function Signature:
        a. C++: void reverse();
        b. Java: void reverse();
        c. Python: def reverse(self):
    - **Description:** Reverse the order of elements in the list. This operation should modify the list in place so that the last element now becomes the first element, second last becomes the second element and so on in the modified list. When printed by the printList() function it print the list in reverse order.
4. **Add an element at Back**
    - Function Signature:
        a. C++: void addBack(const Ele& e);
        b. Java void addBack(String newElement);
        c. Python: def addBack(self):
    - **Description:** add a new element (newElement) at the back of the list).
5. **Remove an element from Back**
    - Function Signature:
        a. C++: void removeBack();
        b. Java: void removeBack();
        c. Python: def removeBack(self);
    - **Description:** remove the element from the back of the list.

3. **Testing and Output:**
    - Implement a main() function or equivalent (Main class in Java) in a separate test file to test all the newly implemented functions.
    - The main() function should create a DLinkedList object, perform various operations using the new functions, and print the list after each operation to demonstrate the correctness of your implementation.

- Example test sequence:
    1. Add several objects in the doubly linked list.
    2. Find an element in the list
    3. Insert an element after specific elements.
    4. Reverse the list.

Part 2: Score Management System

1. **Starting Code:**
    - You are provided with basic classes GameEntry and Scores that manage a list of player scores using Arrays, Vectors, or Lists.
    - Study the provided code carefully to understand its structure and functionality.
    - The starting code will differ slightly based on the language you choose. Modify as appropriate:
        - C++: Classes with std::vector for dynamic arrays.
        - Java: Classes with ArrayList.
        - Python: Classes with native list.
2. **Tasks:**

    You need to implement the following additional functions in the Scores class:

1. **Search the Score of a Particular Player:**
    - Function Signature:
        - C++: int searchScore(const std::string& playerName) const;
        - Java: int searchScore(String playerName);
        - Python: def search_score(self, player_name):
    - Description: Search the score of a player by name. Return the score if found else return -1.
2. **Calculate the Average of the Maximum Scores:**
    - Function Signature:
        - C++: double averageMaxScores() const;
        - Java: double averageMaxScores();
        - Python: def average_max_scores(self):
    - Description: Calculate and return the average of all scores stored in the Scores list.
3. **Find the Minimum and Maximum Scores:**
    - Function Signature:
        - C++: std::pair<int, int> findMinMaxScores() const;

        - Java: int[] findMinMaxScores();
        - Python: def find_min_max_scores(self):
    - Description: Find and return the minimum and maximum scores as a pair of integers.

3. **Testing and Output:**
    - Implement a main() function in a separate test file to test all the newly implemented functions.
    - The main() function should create a Scores object, add several GameEntry objects to the scores list, and demonstrate the functionality of the new methods.
    - **Example test sequence:**

      Add several GameEntry objects with player names and scores. Search for the score of a specific player.

      Calculate and print the average score of all players. Find and print the minimum and maximum scores.

      Print the list of all player scores.

Submission Requirements:

Submit the following files for both parts:

1. C++: .h and .cpp files containing class declarations and implementations for both parts.
2. Java: .java files containing class implementations for both parts.
3. Python: .py files containing class implementations and test cases for both parts.

Ensure that your code compiles (for C++ and Java) or runs without errors (for Python).

Grading Criteria:

- Ensure your code is well-organized and properly commented. Use meaningful variable names and follow good coding practices.
- Correctness: The functions should work as specified and handle edge cases (e.g., player not found, empty list).
- Code Quality: Code should be well-structured, properly commented, and follow best practices.
- Testing: The main() function should test the newly implemented functions with a variety of inputs.

Submission Guidelines

Screenshots of Outputs:

Include screenshots of your program outputs in a separate MS Word document to demonstrate the functionality of your code. Show the testing of all the existing and new functions.

Doubly Linked List Program (DLinkedList.h and DLinkedList.cpp):

- You are provided with 5 function signatures in C++, Java and Python

Note: the 'findNode' function is used to locate and return the node corresponding to a given element and will be passed as the first parameter in the following function:

void insertAfter(const Ele& existingElement, const Ele& newElement);

Note: int find(const Ele& element);

This function should return the position of the given element in the list as an integer. The first element in the list should be considered to be at position 1, the second at position 2, and so on.

Testing the Programs:

- For both programs (Doubly Linked List and Score Management), keep the original main() function and modify it as needed for testing purposes. Do not create a new main() function.
- For Python implementation:
  Include all your code in a single file.

Score Management Program:

- **Note:**
  - C++ Users: For the findMinMaxScores() function, it should return a std::pair<int, int> containing the minimum score (as the first value) and the maximum score (as the second value). The pair data structure is part of the C++ Standard Library (std).
  - Python Users:
    Python also has a similar data structure, so ensure your implementation uses the equivalent structure to return the minimum and maximum scores.

Submission Guidelines:

1. **Organize your files into folders as follows:**
   - Doubly Linked List Program: Place all related files along with the screenshots in MS Word file in a folder named 'DLinkedProgram'.
   - Score Management Program: Place all related files along with the screenshots in MS Word file in a folder named 'ScoreManagementProgram'.
2. **Create a separate submission folder:**
   - **The name of this folder should follow the format:**
     XXX_Assignment1_YourName_CIS350,
     where:
     1. **XXX represents the programming language you have chosen.**
     2. **YourName should be replaced with your first and last name.**

For example:

- If you are submitting the assignment in C++, name the folder: CPP_Assignment1_YourName_CIS350.
- If you are submitting the assignment in Python, name the folder: Python_Assignment1_YourName_CIS350.

3. **Bundling and Submitting:**
   - **Copy and paste the 2 program folders (DLinkedProgram, and ScoreManagementProgram) under this folder (XXX_Assignment1_YourName_CIS350)**
   - **Enzip this folder into zip file name, XXX_Assignment1_YourName_CIS350.zip**
   - **Make sure all your code and screenshots are properly organized into the respective folders before submission.**
   - **Correct bundling and submission format will carry 5 points out of 100.**

Please follow the submission guidelines for Assignment 1 based on the programming language you have chosen for your implementation. Ensure you submit the appropriate files for both the Doubly Linked List and Score Management System programs as detailed below.

For C++:

You are required to submit 8 files in total:

1. **Doubly Linked List Program (3 files):**
   - **DLinkedList.h:** This file should contain the class declarations.
   - **DLinkedList.cpp:** This file should contain the class implementations.
   - **DLinkedListTest.cpp:** This file should include the testing of all functions implemented. *(Refer to the attached file for a sample output.)*
2. **Score Management Program (5 files):**
   - **GameEntry.h:** This file should contain the class declarations for the GameEntry class.
   - **Scores.h:** This file should contain the class declarations for the Scores class.
   - **GameEntry.cpp:** This file should contain the class implementations for the GameEntry class.
   - **Scores.cpp:** This file should contain the class implementations for the Scores class.
   - **GameEntryTest.cpp:** This file should include the testing of all functions implemented. *(Refer to the attached file for a sample output.)*

For Java:

You are required to submit 5 files in total:

1. **Doubly Linked List Program (2 files):**
   - **DLinkedList.java:** This file should contain both the class declarations and implementations.
   - **DLinkedListTest.java:** This file should include the testing of all functions implemented. *(Refer to the attached file for a sample output.)*
2. **Score Management Program (3 files):**
   - **GameEntry.java:** This file should contain the class declarations for the GameEntry class.
   - **Scores.java:** This file should contain the class declarations for the Scores class.

- **TestGameScoreManagement.java:** This file should include the testing of all functions implemented. *(Refer to the attached file for a sample output.)*

For Python:

You are required to submit 2 files in total:

1. **Doubly Linked List Program (1 file):**
   - **chap7-doubly-linked-list.py:** This file should contain class declarations, implementations, and testing of all functions.
2. **Score Management Program (1 file):**
   - **chap5-score-mgmt.py:** This file should contain class declarations, implementations, and testing of all functions.

Additional Requirement:

Along with the code files, please also submit screenshots showing the output of your 2 programs for each function you have implemented. These screenshots should demonstrate the working of your programs. *(Refer to the attached file for a sample output.)*

==========

C++ Assignment Rubric: Doubly Linked List & Score Management Programs

Total Points: 100

1. Doubly Linked List Program (40 points)

Criteria	Points	Description
Correctness of find()	5	Correct implementation of the find() function, which returns the position of a given element in the list. The first element is at position 1.
Correctness of insertAfter()	5	Accurate implementation of the insertAfter() function (use the node returned by findNode() as the first parameter as an existing element). Inserts a new element (newElement) after a given element (existingElement). If the existingElement is not found, print a suitable message.
Correctness of reverse()	10	Accurate implementation of the reverse() function which modifies the list and reverses the order of list elements in place.
Correctness of addBack()	5	Accurate implementation of the addBack() function which adds a new element (newElement) at the back of the list.
Correctness of removeBack()	5	Correctness of **removeBack**() function which removes the element from the back of the list
Modification of main() for Testing	10	The original main() is modified correctly to test the new and existing functionalities without creating a new main() function.

2. Score Management Program (40 points)

Criteria	Points	Description
Correctness of findMinMaxScores()	10	Accurate implementation of the findMinMaxScores() function using std::pair<int, int> to return the minimum and maximum scores.
Correctness of averageMaxScores()	10	Correct implementation of the averageMaxScores() function, returning the correct average score for all entries.
Correctness of searchScore()	10	Accurate implementation of the searchScore() function to return the score for a specific player by name.
Modification of main() for Testing	10	The original main() is modified correctly to test the new and existing functionalities without creating a new main() function.

3. Submission Format (5 points)

Criteria	Points	Description
Correct Folder Structure	5	All files are correctly organized into folders named DLinkedProgram and ScoreManagementProgram. The submission folder is named as per instructions.

4. Screenshots (5 points)

Criteria	Points	Description
Correctness and Inclusion	5	Screenshots clearly showing program output are included in a separate MS Word document, covering all tested functions for both programs.

5. Thorough Testing (10 points)

Criteria	Points	Description
Test Coverage	10	Tests cover all key functions, including existing and find(), reverse(), insertAfter(), addBack(), removeBack() for Doubly Linked List, and findMinMaxScores(), averageMaxScores(), searchScore() for Score Management. Tests include edge cases (e.g., searching for an element not in the list).

Grading Summary:

- Doubly Linked List Program: 40 points
- Score Management Program: 40 points
- Correct Submission: 5 points
- Correct Screenshots: 5 points
- Thorough Testing: 10 points

Sample Outputs

Below are the sample outputs (**not complete**) for each program.

Note: Your program output will cover all the new and existing functions testing (not all shown in the below screenshots).

Sample Output for Doubly Linked List Implementation:

```
Adding elements to the list: SFO -> PVD -> JFK

Inserting "BWI" after "PVD":
SFO -> PVD -> BWI -> JFK

Reversing the list:
JFK -> BWI -> PVD -> SFO

Finding "PVD":
"PVD" found at position 3
```

Another Sample:

Is the list empty: True

adding elements to the list: SFO -> PVD -> JFK

original list

SFO

PVD

JFK

Front:SFO

Back: JFK

finding element PVD in the list

PVD found in the list at position 1

list before adding BWI after PVD

SFO

PVD

JFK

list after adding BWI after PVD

SFO

PVD

BWI

JFK

Reversing the list

JFK

BWI

PVD

SFO

print all elements in reverse order

SFO

PVD

BWI

JFK

Is the list empty: False

list before removing front

JFK

BWI

PVD

SFO

list after removing front

BWI

PVD

SFO

list after removing back

BWI

PVD

Is the list empty: False

Part 2: Score Management System

```
Adding player scores:
Mike : 1105
Rob : 750
Paul : 720
Anna : 660
Rose : 590
Jack : 510
Jill : 740

Searching for Rob's score:
Rob's score is 750

Calculating the average score:
Average score: 725.0

Finding minimum and maximum scores:
Minimum score: 510
Maximum score: 1105

List of all player scores:
Mike : 1105
Rob : 750
Paul : 720
Anna : 660
Rose : 590
Jack : 510
Jill : 740
```

Another Sample:

Num of Entries: 7

Mike : 1105

Rob : 750

Paul : 720

Anna : 660

Rose : 590

Jack : 510

Jill : 740

Rob's score is: 750

Average score: 725

Minimum score: 510

Maximum score: 750

After removing entry at index 3:

Mike : 1105

Rob : 750

Paul : 720

Rose : 590

Jack : 510

Jill : 740