CSCI207: Data Structures

Assignment 1

Requirements

1. Apply Insertion sort on a singly linked list.
2. Remove elements from a doubly linked list.
3. Apply Insertion sort on an Array.

Objectives

1. Practice on inheritance.
2. Practice on virtual functions as they have to override the insert method.
3. Compare the complexity of handling insertion in lists implemented as linked lists **versus** those of implemented as arrays.

Insertion sort on a singly linked list

- Given the linked list data structure discussed in the lecture, implement a sub-class **TSortedList** that makes sure that elements are inserted and maintained in a descending order, as per the alphabetic ordering, in the list. So, given the input sequence {“orange”, “apple”, “banana”, “zebra”}, when printing the list after inserting the last element it should print like {“zebra”, “orange”, “banana”, “apple”}.

Note:
That with inheritance:

- You have a parent class called “**Node**” that contain [data, *next] and a sub class called “**TSortedList**”
- You have to only care about the insertion situation as deletion should still be handled by the parent class.
- In the basic implementation, assume that you only have ***head*** which is a pointer to the head of the list. Now, if we have both ***head*** and ***tail*** pointers, what improvements can bedone to speed up the insertion?

Remove elements in a doubly linked list

- Given the doubly linked list data structure discussed in the lecture, implement a sub-class **DuplicateManipuationList** that has a new function for **removing duplicates from** a list.

- Add a new method to your **DuplicateManipuationList** called **removeDuplicates(…)**. The method takes as an argument called **ListToRemove** that is a singly linked list that contains some values. The method should scan the linked list for each member of the **ListToRemove,** and remove **all occurrences (not only the duplicates)** of that member. Besides updating the DuplicateManipulationList, the **removeDuplicates(…)** function should return a new list that shows the number of duplicates for each deleted member.

Note:
That with inheritance:
- You have a parent class called "**Node**" that contain [data, *next, *prev] and a sub class called "**DuplicateManipuationList**"

For example:

- Assume that the **DuplicateManipulationList** initially contains the values {7,15, 3, 2, 1, 3, 7, 15, 15}, and the argument **ListToRemove** contains {7, 15}. After invoking the removeDuplicates(**ListToRemove**), the **DuplicateManipulationList** would contain the values {3, 2, 1, 3}, and the function would return a list containing the values {2, 3}, where 2 indicates that **7** had two occurrences in the list, and **15** had three occurrences in the list. Note that the above test values are **not** the only values that you should consider.

Insertion sort in an array (Bonus)

- In this assignment, we need to have our list implemented as a normal Array [neither vector nor Linked list] **TSortedArrayList**. So,we already have all elements contiguous in memory. Yet, we have to take care of adding a new element to a full array. In this case, we have to allocate a larger array and copy elements from the old array to the new one, delete the old array and then insert the element in the new array.

- Also, with insertion sort, we mean that we always keep the array sorted in a descending order, thus, we have to scan the array to determine the insertion position, then if the insertion position is in the middle of the array, we have to move the elements to the right of the insertion position (those of smaller or equal value) on cell to the right. This will makeroom for the new element.

- For deletion, we have to move elements to right of the deleted element one cell to the left.Another strategy is to just mark the cell as deleted and do not actually move anything around to save time. Yet, when inserting new elements, we have to account for whether the cell content is actual or deleted.

Due Date: Saturday the 20th of November 2021 at 11:55 PM
Delay Policy: Any delay you lose full assignment grade as the submission will be suspended on the deadline.
Submissions: will be on Moodle not by email.
NOTE: Submit your assignment individually (no teams are permitted). If plagiarism is **DETECTED** or **SUSPECTED**, tough actions will be applied.