Data Structures & Algorithms *COURSE* & *LAB* – Spring 2022

(BS-IT-F20 Morning & Afternoon)

Assignment # 3

Assigned on: ***Wednesday, June 01, 2022***

Submission Deadline: ***Tuesday, June 07, 2022 (till 11:00 PM)***

Instructions:

- **This is an individual assignment. Absolutely NO collaboration is allowed. Any traces of plagiarism/cheating would result in an "F" grade in this course.**
- Do **NOT** copy even a single line of code from any other person or book or Internet or any other source.
- This assignment needs to be submitted in **Soft** form. The **Submission Procedure** is given below.
- Late submissions will NOT be accepted, whatever your excuse may be.
- This Assignment will be counted towards your *Sessional marks* in both **DSA-Course** and **DSA-Lab**.

Submission Procedure:

i. Create an empty folder. Put all of the ***.CPP*, .H, and input file(s)** of your assignment in this folder. Do NOT include any other files in your submission, otherwise your submission will NOT be graded. Don't forget to **mention your Name, Roll Number and Section in comments at the top of each CPP file**.

ii. Now, compress the folder (that you created in previous step) in **.RAR** or **.ZIP** format. The name of the RAR or ZIP file should be ***exactly*** according to the format:

`Mor-A3-BITF20M123` (for Morning section) OR

`Aft-A3-BITF20A456` (for Afternoon section),

where the text shown in **BLUE** should be your **complete Roll Number**.

iii. Finally, submit the (single) **.RAR** or **.ZIP** file through **Google Classroom**. Make sure that you press the **SUBMIT** button after uploading your file.

Note: *If you do not follow the above submission procedure, your Assignment will NOT be graded and you will be awarded ZERO marks.*

These *good programming practices* will also have their (significant) weightage in the marking of this assignment:

- Comment your code intelligently. **Uncommented code will NOT be given any credit.**
- Indent your code properly.
- Use meaningful variable and function names. Use the **camelCase** notation.
- Use meaningful prompt lines/labels for input/output.

If your program **does NOT compile correctly** or it **generates any kind of run-time error** (dangling pointer, memory leak etc.) you will get **ZERO marks** in the whole assignment.

TASK # 1.1 (10 Marks)

In this task, you are required to implement a **singly linked list** class which stores integers in **unsorted** order. Your class declarations should look like:

```
class LinkedList;
class Node {
      friend class LinkedList;
  private:
      int data;
      Node* next;
};
```

```
class LinkedList {
  private:
      Node* head;
  public:
      LinkedList();       // Default constructor
      ~LinkedList();      // Destructor
      // Other member functions are explained below
};
```

Apart from the *default constructor* and *destructor*, the class `LinkedList` class should have the following member functions:

`bool insert (int val)`

This function should insert a new value (`val`) into the linked list, and return **true**. The time complexity of this function should be **constant** i.e. $\boldsymbol{O(1)}$.

`void display ()`

This function should display the **contents** of the linked list on screen. This function should also display the **total number of nodes** present in the linked list.

`bool remove (int val)`

This function removes the node containing the **first occurrence** of `val` from the linked list. This function should return **true** if such a node is found (and removed) from the linked list, and it should return **false** otherwise.

`bool search (int val)`

This function searches for `val` in the linked list. It should return **true** if `val` is present in the linked list and it should return **false** otherwise.

TASK # 1.2 (20 Marks)

Now, you are required to implement the following two member functions of the `LinkedList` class:

`void sort ()`

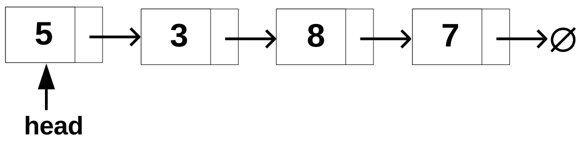
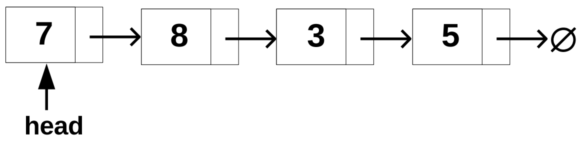
This function sorts the nodes of the linked list in **ascending order** (w.r.t. the "data" present in each node). Keep the following things in mind when implementing this function:

- **When sorting the linked list, you are NOT allowed to modify the "`data`" of any node. You are also NOT allowed to create any new node. In other words, you have to modify links (`next` pointers) of nodes to sort the list.**
- You MUST use a modified version of **Selection sort** to sort the nodes of the linked list. In class, we have already seen how Selection sort works on arrays. Here, you have to implement a slightly modified version of the same algorithm on a linked list. It will work as follows:
  - In the 1st iteration, find the smallest node in the linked list, and move this node (do NOT swap nodes) to the start of linked list.
  - In the 2nd iteration, find the smallest node in the remaining portion of the linked list, and move (place) this node after the smallest node.
  - and so on...
- You are NOT allowed to create any temporary array (or linked list or any other data structure) to perform sorting.

`void reverse ()`

This function should reverse the linked list, iteratively. Keep the following things in mind when implementing this function:

- **You are NOT allowed to modify the "`data`" of any node. You are also NOT allowed to create any new node. So, the reversal is to be done by modifying the links (`next` pointers) of the nodes in the linked list.**
- You are NOT allowed to create any temporary array (or linked list or any other data structure) to perform the reversal.
- The reversal is to be done in **ONE** traversal/pass (going from first node to the last node) through the linked list. You are NOT allowed to traverse the linked list more than once.

For example, if the linked list **before** calling the **`reverse()`** function is: 	Then, **after** the execution of the **`reverse()`** function the linked list should be: 

Note: For the above two functions, use some sample linked lists to see how these functions should work. In other words, **do a lot of paperwork before writing the code**. Also, make sure that all the **boundary cases / special cases** are properly handled.

TASK # 1.3 (10 Marks)

In this task, you are required to write a menu-driven program which allows the user to use all of the aforementioned functionalities. The menu and a sample run of your program should look like as shown below (text shown in **red** is entered by the user):

```
       1. Insert values
       2. Remove a value
       3. Search a value
       4. Display the Linked List
       5. Sort the Linked List
       6. Reverse the Linked List
       7. Empty the Linked List
       8. Exit

Enter your choice: 1
Enter the values to be inserted: 5 2 4 6 8 2 3

Enter your choice: 4
The list contains following 7 elements: 3 2 8 6 4 2 5

Enter your choice: 3
Enter the value to be searched: 8
The number 8 is present in the linked list!

Enter your choice: 5
The list has been sorted!
The list contains following 7 elements: 2 2 3 4 5 6 8
```

```
Enter your choice: 6
The list has been reversed!
The list contains following 7 elements: 8 6 5 4 3 2 2

Enter your choice: 1
Enter the values to be inserted: -6 10

Enter your choice: 4
The list contains following 9 elements: 10 -6 8 6 5 4 3 2 2

Enter your choice: 2
Enter the value to be removed: -6
-6 has been removed from the list

Enter your choice: 2
Enter the value to be removed: 15
ERROR: 15 is not found in the linked list

Enter your choice: 4
The list contains following 8 elements: 10 8 6 5 4 3 2 2

Enter your choice: 7
All the values have been removed from the linked list!

Enter your choice: 4
Linked list is empty!

Enter your choice: 1
Enter the values to be inserted: 1 5 7

Enter your choice: 4
The list contains following 3 elements: 7 5 1

Enter your choice: 8
Bye bye!!
```

Good programming practices (10 Marks)

Note that: These *good programming practices* will also have their (significant) weightage in the marking of this assignment:

- There should be no memory leaks, dangling pointers, or any other type of runtime error in your program.
- **Comment your code** intelligently. *Uncommented code may not be given any credit.*
- Use meaningful variable and function names.
- Indent your code properly.
- Do NOT use any global or static variables.

Moreover, if your submitted program gives an error or a warning message at the time of compilation, you will get a ZERO in the assignment.

☺ GOOD LUCK! ☺