DSA **Lab 6**

Question 1

1. Implement a template class **‘Node’** that contains two data members: A template variable ‘data’ and a Node pointer ‘next’. You may define any member functions, if required, for the template class. **(Implemented in previous lab.)**
2. Now using the above class, implement a template class singly linked **list** with **head** and **tail pointers** which supports the following operations **(Implemented in previous lab using head pointer only)**:
   a. Insert at start `void insertAtStart(Tconst element); Time complexity= O(1)`
   b. Insert at end `void insertAtEnd(Tconst element); Time complexity= O(1)`
   c. Copy constructor `linkedlist(const linkedlist &old_obj);`
   d. Print `void print() const;`
   e. Search an element `bool search(T const& element) const;`
   f. Check whether the list is `empty` `bool isEmpty() const; Time complexity= O(1)`
   g. Insert value v1 before value v2 `bool insertBefore(T const v1, T const v2) const;`
   h. Delete all occurrences of a given value `void deleteAll(T constvalue)`
   i. Destructor
   j. Reverse Print `void reversePrint();`
   k. Delete from Start `void DeleteAtStart();`

   Note: You should update your previous methods by adding tail pointer.

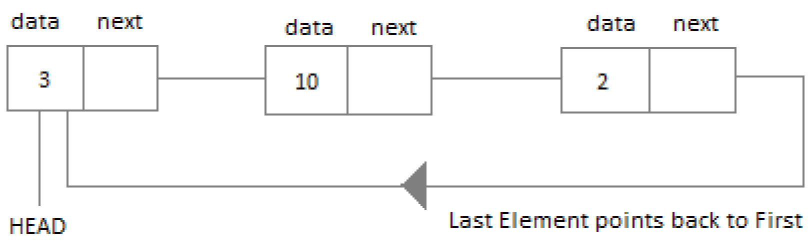
3. Overload following operators:
   1) +=
   2) []

   A general operator overloading method is like this:
   Classname operator + (Classname const &obj);

4. Now create a main function which has the following instructions:
   a. Define a linked list object of type **int**.
   b. Insert 2, 6, and 7 at start
   c. Insert 9 at the end.
   d. Now insert 7, 8, and 9 at start.
   e. Delete all occurrences of 7.
   f. Now print the linked list.
   g. Search for 2, 9 and 10.
   h. Now delete from Start and print the linked list.

Q # 2: Implement a template class **circular** singly linked **list** with **head pointer only**. (Modifications in your link list implemented in previous lab)

Note: You can use a dummy node and handle all checks or test cases in all methods.


data
next
3
data
next
10
data
next
2
HEAD
Last Element points back to First