DATA STRUCTURES

Fall 2023

LAB 05

Learning Outcomes

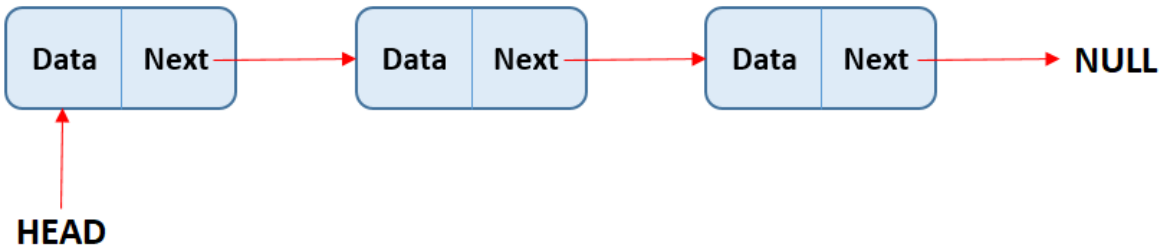
In this lab you are expected to learn the following:

- Singly Linked List

LINKED LIST

Linked List is a linear collection of data elements whose order, unlike array, is not given by their physical placement in memory. Instead, each element points to the next. This data structure consists of a collection of nodes which together represent a sequence. Linked List data structure is not size bound and the size increases dynamically.

The Linked List is accessed by a reference called **head** which always points to the first node of Linked List. The last node of the Linked List is indicated by its **next** pointing to **null**.



TASK 1:

Implementation will have three classes:

- Employee
- Node
- LinkedList

Each **Node** of a linked list will have two data members:

- Record of an **Employee**
- A pointer to the next node.

And it will only have 1 function i.e., default constructor.

Following Table shows the list of ~~**Employee Class**~~ data members:

Data Member	Data Type
Emp ID	int
NIC	string
Salary	Int/double
Bonus	Int/double

Write appropriate member function for **Employee Class**

- Default and Parameterized Constructors

- Display function.

Singly Linked list after five insertions:

Operations in LinkedList Class

1. LinkedList () (12 mins)

   Requirements:

   None

   Results:

   Constructor. Creates an empty list.

2. ~ LinkedList () (12 mins)

   Requirements:

   None

   Results:

   Destructor. Deallocates (frees) the memory used to store a list.

3. void insert (const Employee &emp) (15 mins)

   Requirements:

   None

   Results:

   If the list is not empty, then inserts **emp** at the end.

4. void remove (int id) (15 mins)

   Requirements:

   List is not empty.

   .

Results:

Removes the data item whose employee id matched the parameter id. If the id doesn't exist display amessage "Record does not exist

5. void UpdateSalary (const int &salary, int id) (15 mins)

   Requirements:
   List is not empty.

   Results:
   Locate the data item whose employee id matches the parameter id, then update the respective data itemsalary member with the parameter salary.

6. void UpdateBonus (const int &bonus, int id) (15 mins)

   Requirements:
   List is not empty.

   Results:
   Locate the data item whose employee id matches the parameter id, then update the respective data itembonus member with the parameter bonus.

7. void clear () (12 mins)

   Requirements:
   None

   Results:
   Removes all the data items in a list.

8. bool isEmpty () (12 mins)

   Requirements:
   None

   Results:
   Returns **true** if a list is empty. Otherwise, returns **false**.

9. void display () (12 mins)

   Requirements:
   List is not empty.

   Results:
   Display the data items separated with two empty lines.

10. void sort () (15 mins)

    Requirements:
    List is not empty.

    Results:
    Display the sorted data items on the basis of the employee salary.

    Note: Write the code to sort the nodes on the basis of their data items. i.e., **do not** copy the dataof nodes to each other.

11. int main() (15 mins)

```
{
        List l1;
        cout << l1.isEmpty() << endl;
        Employee E1 (id, nic, salary, bonus);
        Employee E2 (id, nic, salary, bonus);
        Employee E3 (id, nic, salary, bonus);
        l1.insert(E1);
        l1.insert(E2);
        l1.insert(E3);
        l1.display();
        l1.remove(2);
        l1.display();
        l1.UpdateSalary(200, 2);
        l1.UpdateBonus(150,1)
        l1.display();
        cout << l1.isEmpty();
        cout << endl;
        l1.sort();
        l1.display();
        return 0;
}
```

Sample Input:

Emp1	
EmpID	18i-8079
NIC	12345-1234567-1
Salary	50,000
Bonus	5000

Emp2	
EmpID	18i-8098
NIC	12355-1034577-0
Salary	30,000
Bonus	0

Emp3	
EmpID	18i-4243
NIC	10045-1200567-1
Salary	70,000
Bonus	10000

Emp4	
EmpID	18i-8843
NIC	10845-1200567-1
Salary	100,000
Bonus	15000

Result