|  |  |
| --- | --- |
| **Name** |  |
| **CMS ID** |  |
| **Date** |  |

**Objectives:**

After completing this Lab students will able to

1. Understand the concept and usage of Linked Lists in programming.
2. Implementation of Dictionary Operations on Linked List.

**Objective 1: *Understanding Linked Lists.***

A linked list is an ordered and simple linear data structure which can change during execution.

* Successive elements are connected by pointers.
* Last element points to NULL.
* It can grow or shrink in size during execution of a program.
* It can be made just as long as required.
* It does not waste memory space.
* The nodes do not reside in sequential locations.
* The locations of the nodes may change on different runs.

A linked list is a complex data structure, especially useful in systems or applications programming. A linked list is comprised of a series of nodes, each node containing a data element, and a pointer to the next node, eg,

bat

cat

sat

vat

null

A structure which contains a data element and a pointer to the next node is created by,

struct list {

int value;

list \*next;

};

**Array versus Linked Lists:**

* Arrays are suitable for:
  + Inserting/deleting an element at the end.
  + Randomly accessing any element.
  + Searching the list for a particular value.
* Linked lists are suitable for:
  + Inserting an element randomly.
  + Deleting an element randomly.
  + Applications where sequential access is required.
  + In situations where the number of elements cannot be predicted beforehand.

**Objective – 2: *Implementing of Linked List.***

**Basic Operations on a List**

* Creating a list
* Traversing the list
* Inserting node in the list
* Appending the list
* Searching within list
* Deleting an item from the list
* Counting nodes in a list

**Creating a List**

To start with, we have to create a node (the first node), and make head point to it.

**Traversing a List**

Once the linked list has been constructed and *head* points to the first node of the list,

* Follow the pointers.
* Display the contents of the nodes as they are traversed.
* Stop when the *next*pointer points to NULL.

**Task 1.** For implementation of linked list, you need to create a structure that can be named as **List** as its single object represents a single linked list. With the help of source codes provided in 4th lecture, you are supposed to implement the following functions.

1. Creating a list
2. Traversing the list
3. Inserting node in the list
4. Searching within list
5. Deleting an item from the list
6. Counting nodes in a list

**Task 2.** Create two linked lists (two objects of list class/structure) named ListA and ListB, and store the following data in them

17

25

18

14

28

13

Merge the two lists in such a way that they represent a single list such as shown below

18

25

17

13

28

14

**Task 3.** Create a linked list that stores the following information of the employee

1. Name
2. Age
3. Year of joining
4. Create a member function named **insertNode** of List class that Insert the data of following employees in ascending order with respect to the year of joining.

|  |  |  |
| --- | --- | --- |
| Name | Age | Year of joining |
| Nadir Hussain | 18 | 2019 |
| Bakhtawar | 19 | 2015 |
| Javeria | 105 | 1987 |
| Nageeta | 95 | 1990 |
| Karishma | 115 | 2011 |
| Shakeel | 10 | 2019 |
| Malook | 15 | 2017 |

1. Create another function that when called, delete the data of the person with maximum experience in order to fulfil the policy of company to expel the oldest employee when it wants to downsize. The function should display all the relevant information of employee before deleting.
2. Create searchName function that takes name as parameter and display all the relevant information about that person.
3. Create another function named searchAge that takes age as parameter and display information of all employees with the given age.