**Name: Abdul Ghaffar Kalhoro**

**Registration# 194699**

**Class: BSCS-6c**

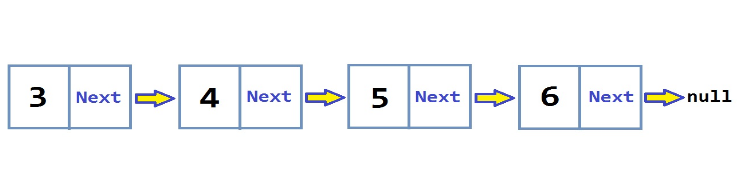
**Course: Data Structures and algorithms**

**Date: 13-Oct-17**

**Lab5: Writing details about the linked list and provide pseudo code for the different linked list operations.**

Question 1:

**Singly Linked List with Visual Representation:**

Linked list is one of the common data structure in programming. It is a linear data structure that uses the pointers to link the objects or nodes together. A node in a linked list contains user defined data (I-e numbers, names etc.) together with this data we have a pointer variable that is used to point the address of next node, so when we create different nodes then by pointing the previous one to the next one we create a linked list, that is nothing but the collection of nodes pointing to each other. The very basic structure of linked list is provided below: 

In above visualization, 4 nodes are shown with two different boxes each, the left part of the node represents the data box and the right shows the stored address of the next node.

We can use the linked list instead of the arrays, which are also the data structures but the linked list has number of benefits over arrays like: we can insert or delete elements in linked list very easily as compared to that of arrays. But when we talk about the searching of any element, in that case the arrays have better use than linked list, as you only need to search the index position of that element to be searched with the complexity of O(1), but in linked list we need to traverse the whole linked list in order to reach our desired element stored in it and sometimes its complexity can be O(n).

# **Advantages of linked list:**

* It is a dynamic data structure, which can be increased and decreased while the program is running.
* The initial size for the linked list is not defined and is not fixed.
* The data can be added or removed from the middle of list.
* The dynamic data structures like stacks and queues can be implemented using a linked list.

# **Disadvantages of linked list:**

* As the linked list elements are sequential data structure so it is required to start from 1st element to reach the other elements
* More memory is used than that of arrays due to the storage used by their pointers
* Reversing the linked list is difficult as we cannot traverse back in a singly linked list.

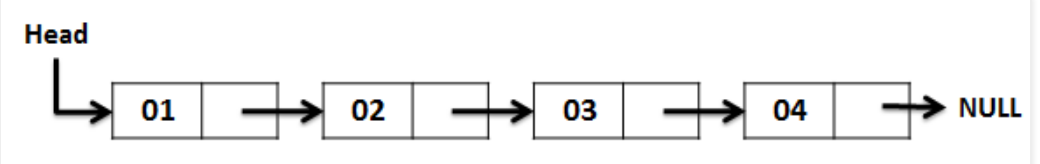
Question 2:

# Operation performed using linked list:

## Creation:

The linked list can be created using the C++ programing language.

As C++ provides Object Oriented Programming concept so we can create linked by making the class of node and use it in the other class named ‘list’ and use the objects of the node class for creating the list.



**pseudo code:**

Create: struct node

start node

declare varibles: ‘data’ as integer

and ‘next’ as node type variable

end node

make class named ‘list’

start list

declare private variables:

node types: head and tail

public instances:

in the ‘list constructor’:

initialize as;

head = null

tail = null

declare a void return type function “CreateNode(value: integer)”

start function:

declare a node type variable “temp”

set: temp data = value

temp next = null

put a condition

:

if head is equal null:

initialize both head and tail equal to temp

and the make temp equals nulll

:

else temp in temp next

and make tail equals temp

end else

end function

end list

## Display:

**pseudo code:**

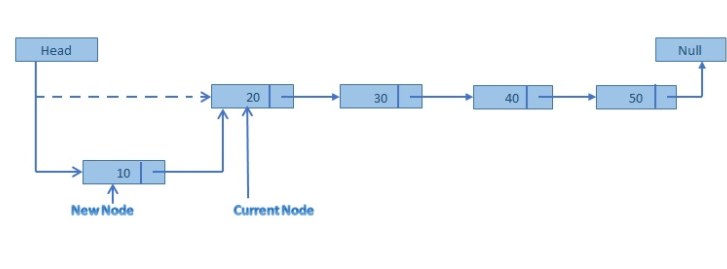
* Make a display function with no return type and empty parameter
* Create a variable type node named “temp”
* Set temp equals head
* Start a loop with condition as temp not equals null
* Then Print temp data
* Set temp equals ‘temp next’
* End loop
* End function

## Insertion:

Insertion is a function which is used to insert any data in the created linked list by adding a new node in it. There are three ways of inserting the node in the linked list, which are given below:

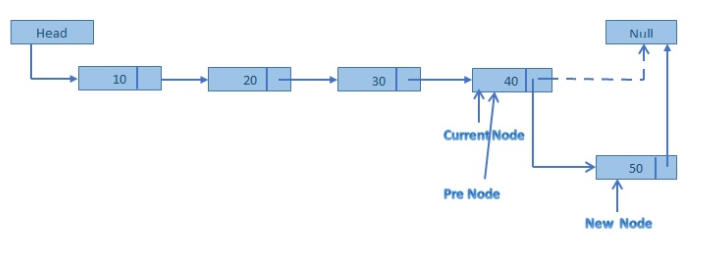
**pseudo code:**

### Insert at start:



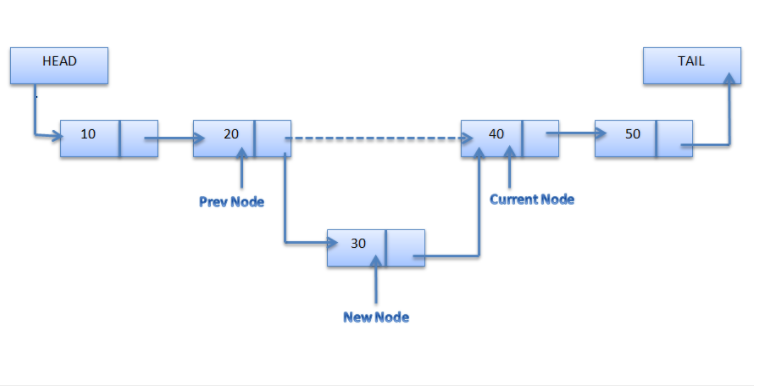
* Create a void return type function “InsertAtStart” with parameter integer type number named “no”
* Create a variable type node named “temp”
* Store integer “no” in temp data
* Set temp next equals head
* And then store temp variable in head.
* End function

### Insert at end:



* Create a void return type function “InsertAtEnd” with parameter integer type number named “no”
* Create a variable type node named “temp”
* Store integer “no” in temp data
* Set temp next equals null
* use if condition and set condition as if head equals equals to null
* then set head equals temp and also make temp as tail in if condition’s body
* else set tail next equals temp
* and store address of tail next into tail
* end condition
* end function

### Insert at specific position:



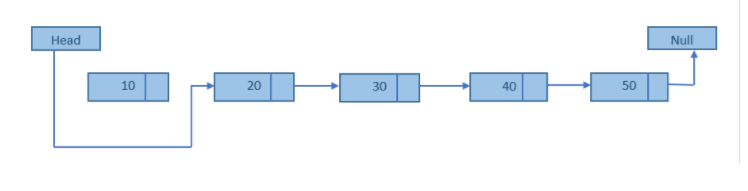
* Create a void return type function “InsertAtSpecific” with two integer type parameters named “position” and “value”
* Create variables type node named “temp”, “current” and “previous”
* Set current equals head
* Set previous equals current and current equals current next in the body of loop with loop counter equals 1 with condition as counter is less than the position.
* Outside the loop body make temporary data equals ‘value’
* Set previous next equals ‘temp’
* And make temp next equals current
* End function

## Deletion:

In Linked list as we can add any node in the created list so we can also delete any node from that list. The different types of deletion are described below:

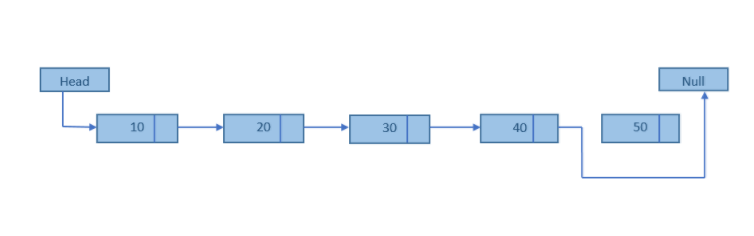
**pseudo code:**

### Delete at start:



* Create a void return type function “DeleteAtStart” with empty parameter.
* Create a variable type node named “temp”.
* Set temp equals head
* And make head to the node stored in the head next
* Delete ‘temp’
* End function

### Delete at end:



* Create a void return type function “DeleteAtEND” with empty parameter.
* Create variable types node named “current” and previous.
* Set current node equals head node
* Write a while loop with condition that current next is not equals ‘NULL’
* In the body of loop,:

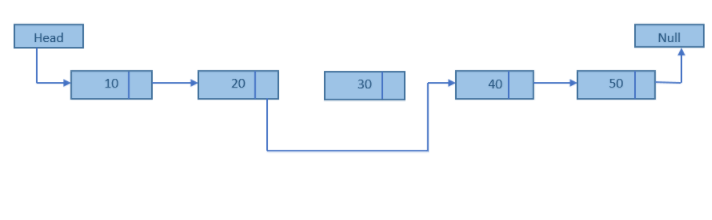
Set:

previous equals to current node

and current equals to ‘current next’

* After loop execution set tail node equals previous and ‘previous next’ equals ‘null’
* Delete current node
* End function

### Delete at specific position



* Create a void return type function “DeleteAtEND” with integer parameter “position”.
* Create variable types node named “current” and previous.
* Set current equals head
* Write loop such that loop counter equals 1 and make counter less than position in loop condition part.
* In the loop body:

Set previous equals current

And current equals ‘current next’

* After loop body set ‘previous next’ equals current next.
* End function

## Count length of list

**pseudo code:**

* Create integer return type function “LengthOFlist” with empty parameter.
* Create variable types node named “current” and previous.
* Also declare an integer variable “length” and set it equal to 1
* Set temp as the head
* Write while loop with condition that: temp equals to the next belonging to the temp
* In the loop body:

Set:

temp node equals to the temp pointing to the next

`And add 1 to the length variable and store the result in it

* After the loop body return length variable
* End function