Data Structures and Algorithms (CS 2001)

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

School of Computer Engineering

4 Credit   *Lecture Note*

Chapter Contents

| Sr #	Major and Detailed Coverage Area	Hrs
3	**Linked List**	8
	Singly Linked Lists and Chains, Representing Chains in C, Polynomials, Sparse Matrix, Doubly Linked Lists, Circular & Header Linked Lists	

Linked List

Linked List is the **linear data structure** which are connected together via links and is a sequence of nodes which contains items and connection to another node. It is the **second most used data structure** after array.

Important Concepts:

❑ **Node –** Each node of a linked list store a data called an **element**.
❑ **Next –** Each Link of a linked list contain a link to next link called Next.
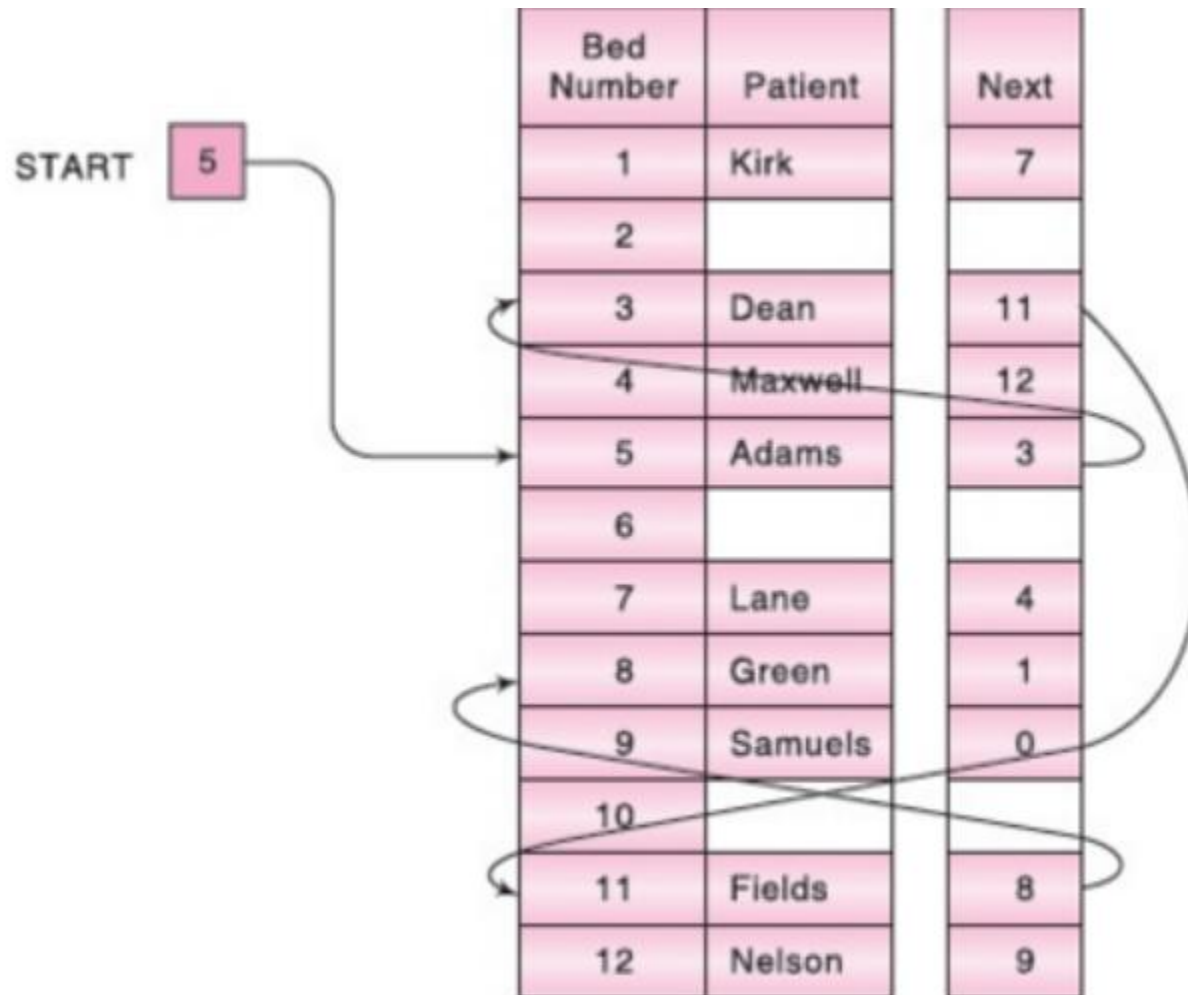❑ **Head** – A Linked List contains the connection link to the first node. This can also be termed as **start**.

Linked list can be visualized as a chain of nodes, where every node points to the next node.

School of Computer Engineering
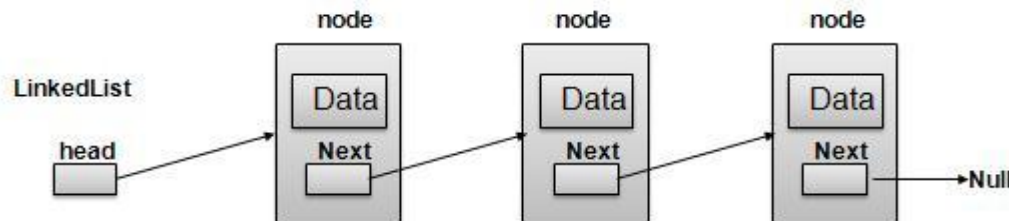
Linked List Example

Linked List Advantages, Disadvantages and Applications

Advantages

- ❑ They are a dynamic in nature which allocates memory when required.
- ❑ Insertion and deletion operations can be easily implemented.
- ❑ Stacks and queues can be easily executed.
- ❑ Linked List reduces the access time.

Disadvantages

- ❑ The memory is wasted as pointers require extra memory for storage.
- ❑ No element can be accessed randomly; it has to access each node sequentially.
- ❑ Reverse traversing is difficult in linked list.

Applications

- ❑ Used to implement stacks, queues, graphs, etc
- ❑ Maintaining directory of names
- ❑ Performing arithmetic operations on long integers.
- ❑ Previous and next page in web browser

Linked List Representation

In memory the linked list is stored in **scattered locations**. The memory for each node is allocated dynamically i.e. as and when required. So the Linked List can increase as per the user wish and the size is not fixed, it can vary.

Suppose first node of linked list is **allocated** with an address 1008. Its graphical representation looks like :

Head
1008

1008
INFO

Suppose next node is allocated at an address 506, so the list becomes

Head
1008

1008
INFO | 506

506
INFO

Suppose next node is allocated with an address 10 and the list becomes

Head
1008

1008
INFO

506
INFO

10
INFO

Address	Comments	Next
e.g. 8005	Root & single node	1008
1008	2nd node insertion	506
506	3rd node insertion	10

School of Computer Engineering
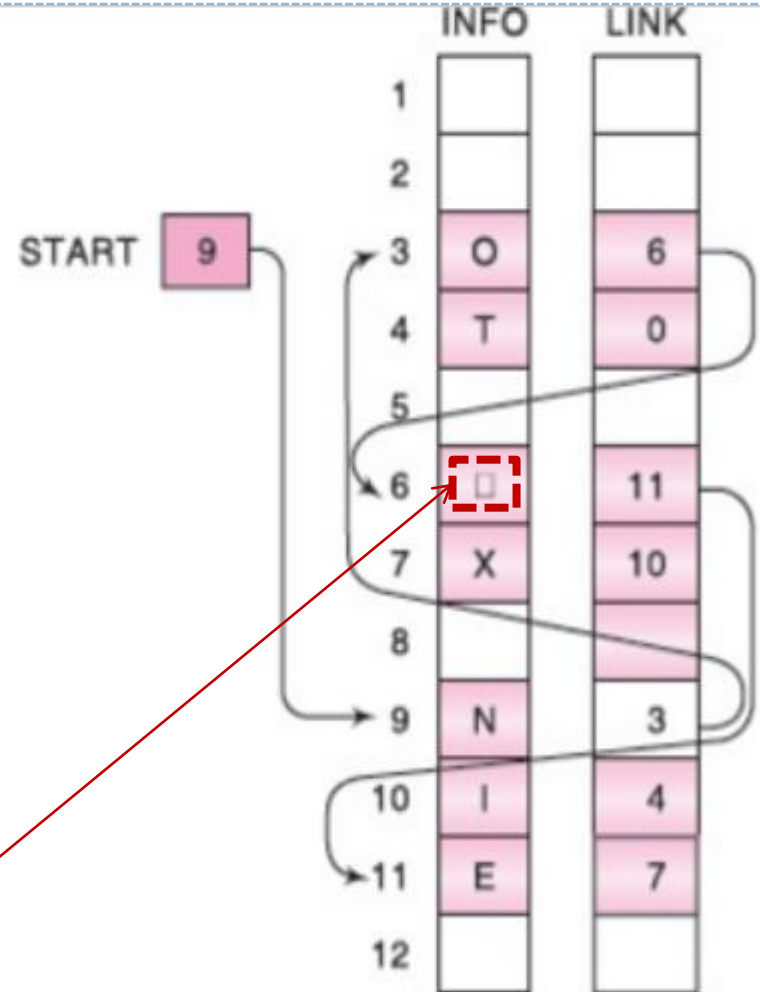
Linked List Representation – structure based

✓ Declaration of struct that should be used to create a list where each node holds info.

struct node
{
    int info;
    struct node *next;
};

✓ The next step is to declare a pointer to serve as the list head, i.e.

struct node *head;

✓ Once node data structure is declared and have created a NULL head pointer, an empty linked list is created.

✓ The next step is to implement operations with the list.

School of Computer Engineering

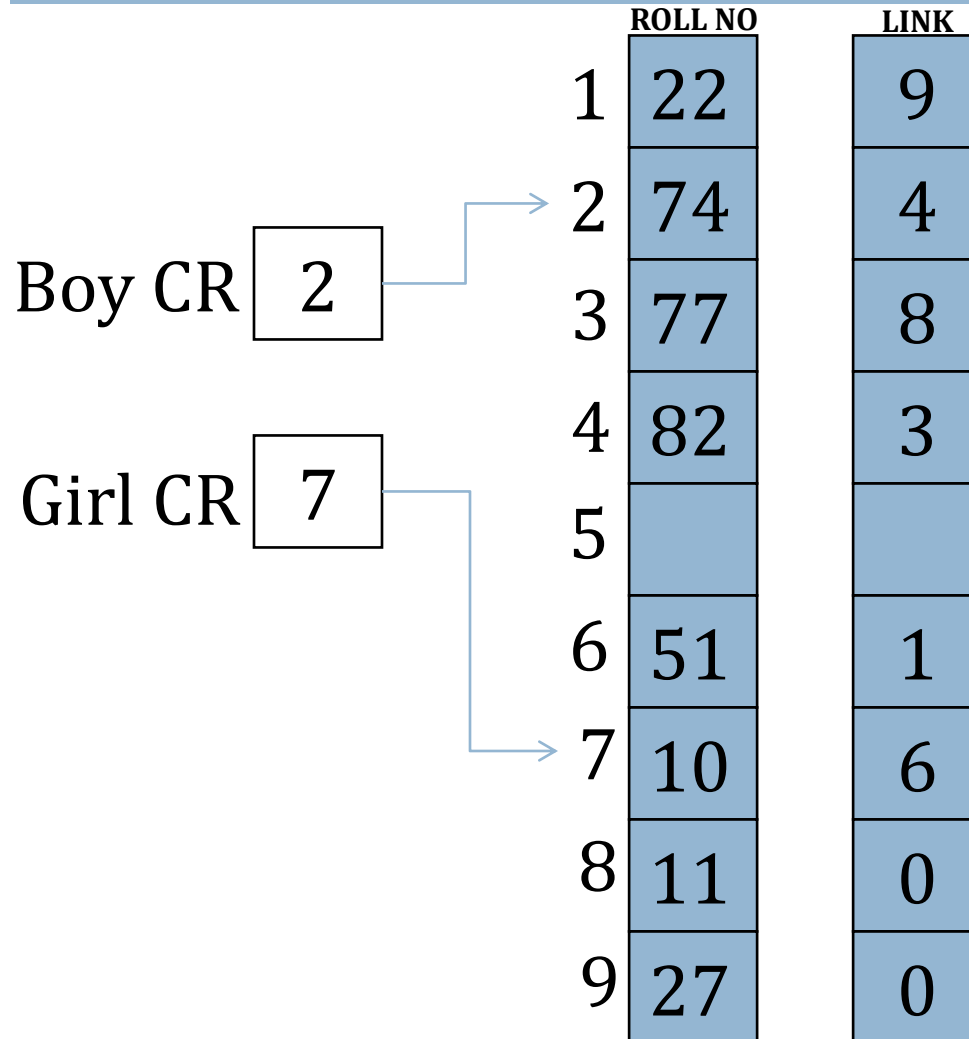# Linked List Representation – Array based

Let **LIST** be the linked list and will be maintained in memory. First of all, LIST requires two linear arrays – call them as **INFO** and **LINK**. INFO[K] represents the **information** part and LINK[K] represents the **next pointer** field of the node of LIST for **Kth** element. LIST also requires 2 variables such as **START** which contains the location of the beginning of the list and the **next pointer sentinel** denoted by **NULL** which **indicates the end of the list**. Since the subscript of the arrays INFO and LINK will usually be positive, so **NULL** should be treated as **0** unless otherwise stated.

Note: represents blank character in the picture

School of Computer Engineering

Example

	ROLL NO	LINK
1	22	9
2	74	4
3	77	8
4	82	3
5		
6	51	1
7	10	6
8	11	0
9	27	0

Boy CR 2
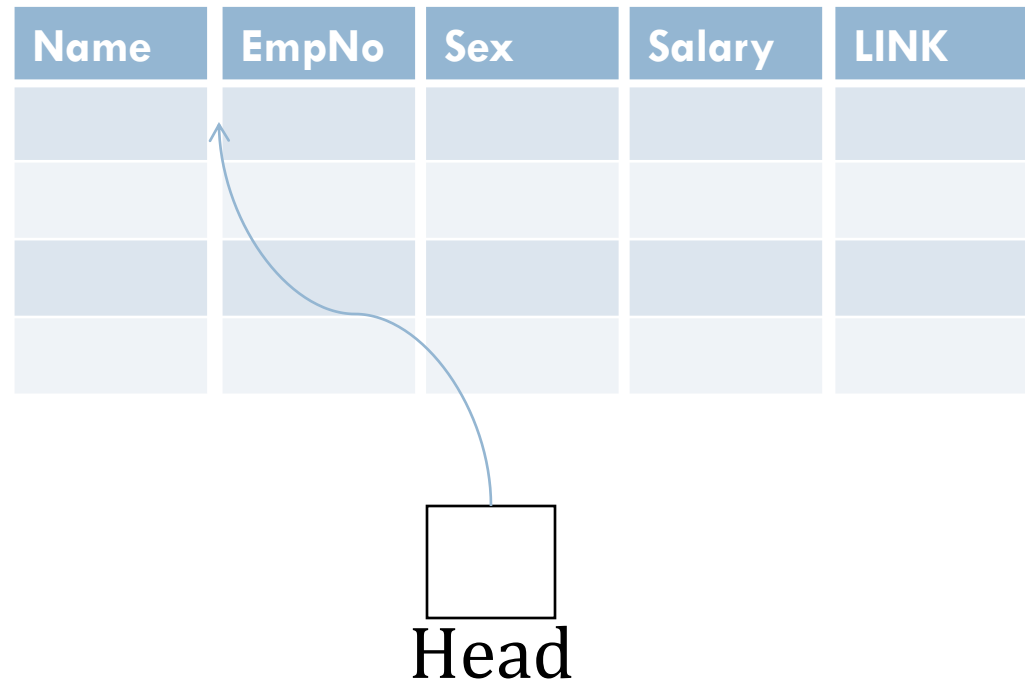
Girl CR 7

School of Computer Engineering

Class Work

Suppose the personnel file of a small company contains the following data of its employees : Name, Employee No, Sex, Salary. Design a linked list using arrays.

Solution

So four parallel arrays say Name, EmpNo, Sex, Salary are required to store the data. Additionally, parallel array LINK is required for the next pointer field of the list and the variable HEAD is to point to the 1st record in the list. 0 can be used to represent the **null** pointer.

Name	EmpNo	Sex	Salary	LINK

Head

Class Work

Suppose the personnel file of a small company contains the following data of its employees : Name, Employee No, Sex, Salary. Design a linked list using structure.

Solution

```
struct personnel
{
    char *name;
    int emp_no;
    char sex;
    float salary;
    struct personnel *next;
}
struct personnel *head;
```
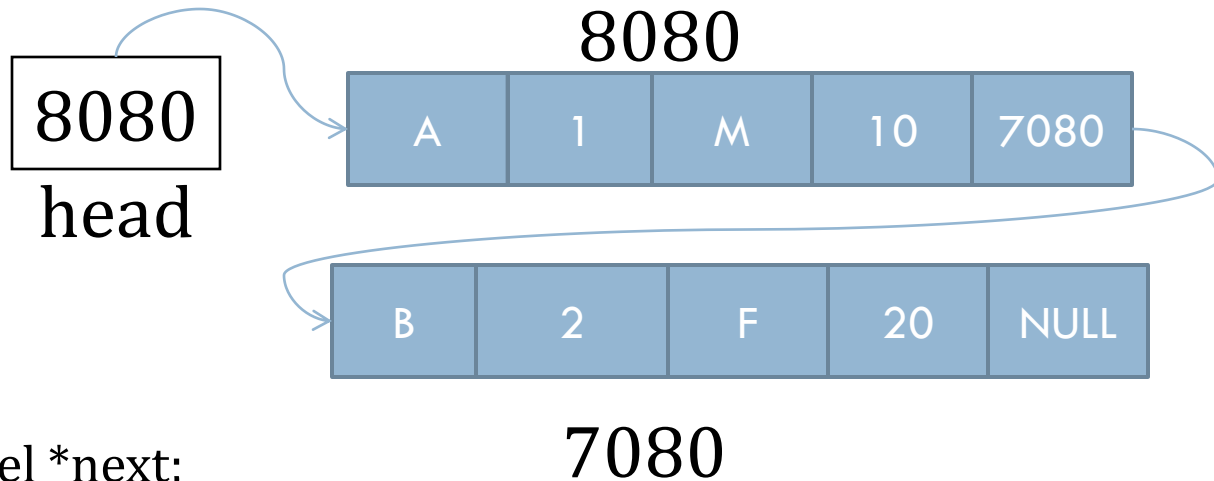
8080

| 8080 | head |

8080

| A | 1 | M | 10 | 7080 |

| B | 2 | F | 20 | NULL |

7080

Types of Linked List

- ❑ **Single Linked List–** Item navigation is forward only. Also called as **one-way** list
- ❑ **Double Linked List –** Items can be navigated forward and backward way
- ❑ **Circular Linked List** – Last item contains link of the first element as next and first element has link to last element as prev.

Single Linked List

Double Linked List

Circular, singly linked list

Circular, doubly linked list
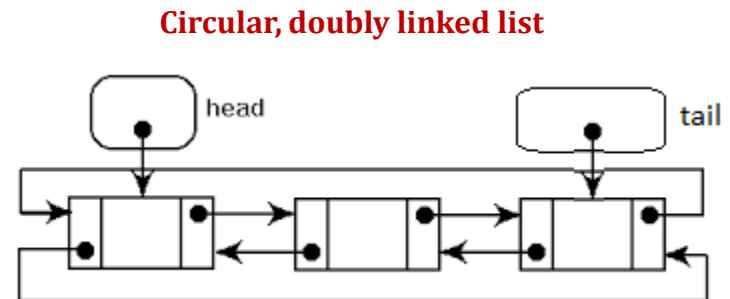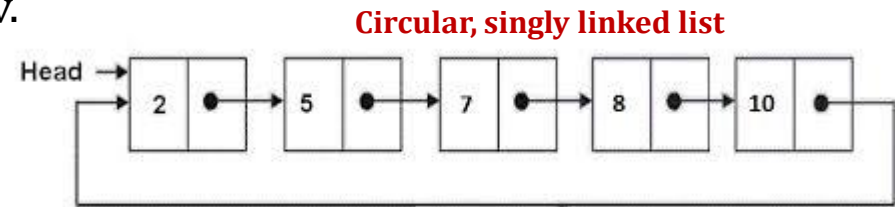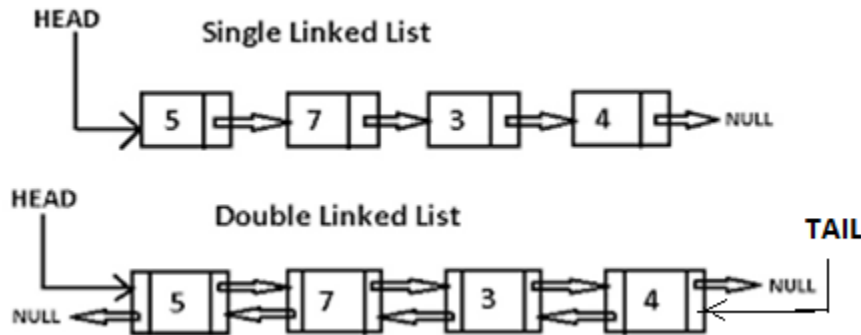
Chains – A chain is a linked list in which each node represents one element
Two-Way Lists – Double and Circular Linked List
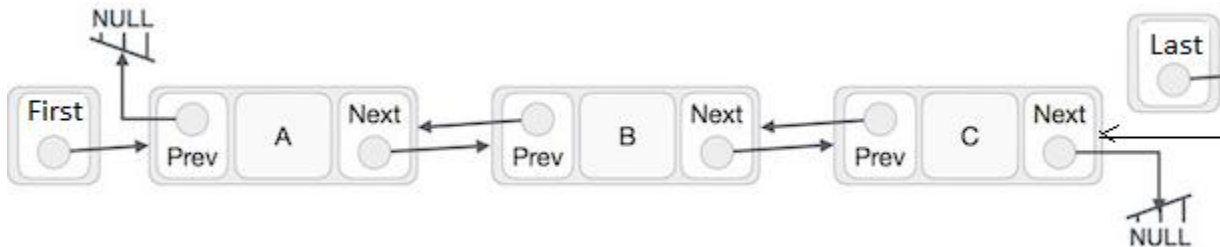
Double Linked List

Doubly Linked List is a variation of Linked list in which navigation is possible in both ways i.e. either forward and backward.

Important Concepts:

❑ **Node –** Each node of a linked list store the data

❑ **Next –** Each node of a linked list contain a link to next link

❑ **Prev –** Each node of a linked list contain a link to previous link

❑ **Linked List –** A Linked List contains the connection link to the first Link called **First** and to the last link called **Last**. **First** is also called as **Head** and **Last** is also called as **Tail.**
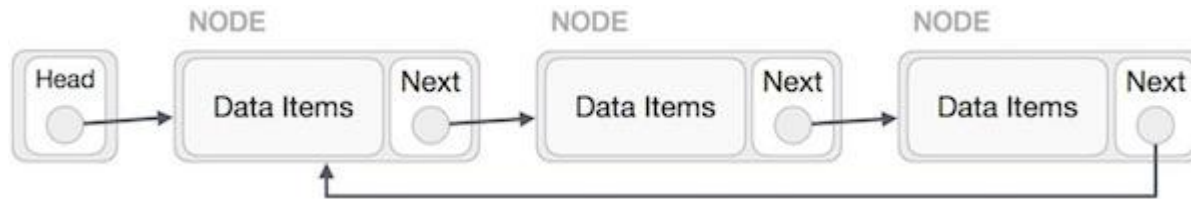
Circular Linked List

Circular Linked List is a variation of Linked list in which first element points to last element and last element points to first element. Both Singly Linked List and Doubly Linked List can be made into as circular linked list.
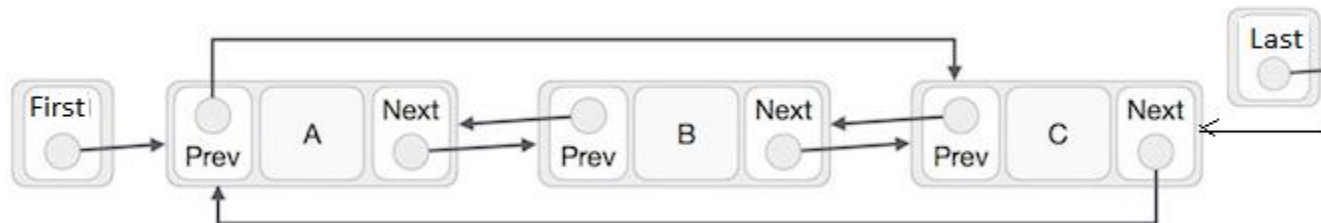
Singly Linked List as Circular

In singly linked list, the next pointer of the last node points to the first node.

Doubly Linked List as Circular

In doubly linked list, the next pointer of the last node points to the first node and the previous pointer of the first node points to the last node making the circular in both directions.

Linked List Basic Operation

❑ **Insertion** – add an element at the beginning of the list.

❑ **Deletion** – delete an element at the beginning of the list.

❑ **Insert Last** - add an element in the end of the list.

❑ **Delete Last** – delete an element from the end of the list.

❑ **Insert After** – add an element after an item of the list.

❑ **Traverse** – Traverse and display the complete list in forward manner.

❑ **Search** – search an element using given key or data item.

❑ **Delete –** delete an element using given key or data item.

❑ **Traverse Backward** – displaying complete list in backward manner. *Applicable for double and circular linked list*

Node Structure

Consider the singly linked list node structure as follows:

struct node

{

    int data;

    struct node *next;

};

struct node *head=NULL;

Create the node structure with data and key

struct node

{

    int data;

    int key;

    struct node *next;

};

struct node *head=NULL;

School of Computer Engineering

Insertion at Start

Inserting a node at start in the SLL (Single Linked List):

1. Create new node
2. Fill data into "data field" of newly created node
3. Attach this newly created node to head
4. Make new node as starting node

Pseudocode:

struct node *new_node;

new_node=(struct node *)malloc(sizeof(struct node));

scanf("%d", &new_node ->data);

if(head==NULL)

  head=new_node;

else

{

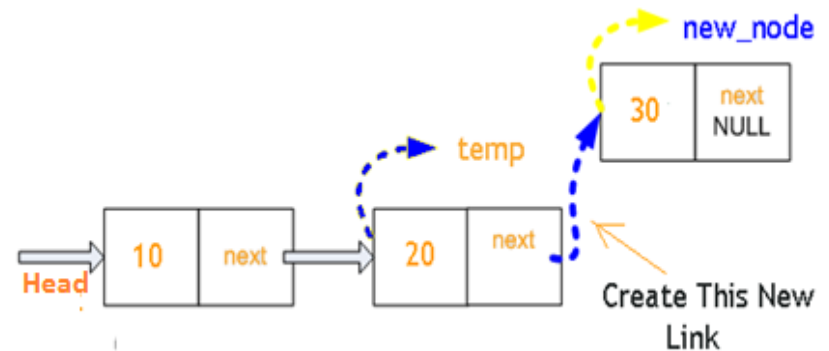 new_node->next=head;

  head=new_node;

}

Insertion at Last

Inserting a node at last in the SLL (Single Linked List):

1. Create New Node
2. Fill Data into "Data Field"
3. Make it's "Pointer" or "Next Field" as NULL
4. Node is to be inserted at Last Position so we need to traverse SLL up to Last Node.
5. Make link between last node and new node

Pseudocode:

```
struct node *new_node = (struct node *)malloc(sizeof(struct node));
scanf("%d", &new_node ->data);
if(head==NULL)
   head=new_node;
else
{
 struct node * temp = NULL;
  for(temp  = head; temp->next!=NULL; temp = temp->next);
  temp->next = new_node;
}
```


School of Computer Engineering

Insertion at Specific Position

Inserting a node at last in the SLL (Single Linked List):

1. Create New Node
2. Fill Data into "Data Field"
3. Make it's "Pointer" or "Next Field" as NULL
4. Node is to be inserted at given Position so we need to traverse SLL up to given position.
5. Make link between specific position node and new node

Pseudocode:

```
struct node *new_node = (struct node *)malloc(sizeof(struct node));
scanf("%d", &new_node ->data);
int pos;
scanf("%d", &pos);
if(head==NULL)
   head=new_node;
else
{
 struct node * temp = NULL; int i=1;
  for(temp  = head; i< pos - 1; temp = temp->next,++i);
  struct node * temp1=temp->next; temp->next = new_node; new_node->next=temp1;
}
```


School of Computer Engineering
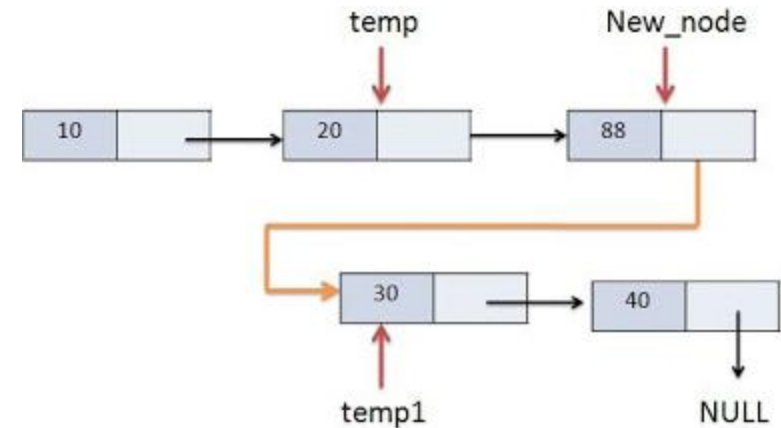
Traversal

Traversing linked list means visiting each and every node of the singly linked list. Following steps are involved while traversing the singly linked list –

1. Firstly move to the first node
2. Fetch the data from the node and if required, perform the operations such as arithmetic operation or any operation depending on data type.
3. After performing operation, advance pointer to next node and perform all above steps on visited node.

Pseudocode:

void traverse()
```
{
    struct node *temp = head; //Move to First Node
    do
    {
      printf("%d", temp->data);
      temp = temp->next; //Move Pointer to Next Node
    }while(temp!=NULL);
}
```

Class Work

Design search algorithm for SLL (Single Linked List)

School of Computer Engineering

Deletion

Pseudocode for Deletion of First Node from Singly Linked List:
struct node *temp;
temp = head;
head = head->next;
free(temp);

Pseudocode for Delete Last Node from Singly Linked List:
struct node *temp = head, t = NULL;
if(head->next==NULL)
 {
  free(head);
  head=NULL;
 }
 else
 {
  for(;temp->next != NULL; temp=temp->next, t = temp);
  free(t->next);
  t->next=NULL;
 }

Class Work

Design an algorithm to delete a node at a specified position.

Design a recursive algorithm to count number of nodes in SLL.

School of Computer Engineering

Linked List Implementation

Singly Linked List

Single Linked List

Double Linked List

Double Linked List

Singly Linked List as Circular

Single Linked List as Circular

Header Linked List

Header linked list is a linked list which always contains a special node called the Header Node, at the beginning of the list. It does not contain actual data item included in the list but usually contains some useful information about the entire linked list. It has two types:

1. Grounded Header Link List– Last Node Contains the NULL Pointer
2. Circular Header Link List – Last Node Points Back to the Header Node.

Note: Start/Head always points to Header Node.

Grounded Header Link List

Circular Header Link List
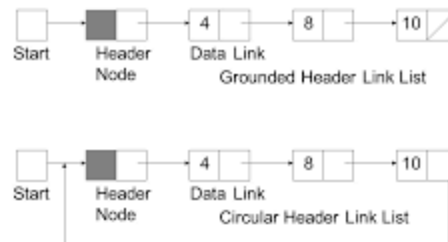
Header Linked List Implementation

Grounded Header Link List

Grounded Header
Link List

C program

Class Work

Write creation and display method of
Circular Header Link List

Polynomials

Polynomial is an expression constructed from one or more variables and constants, using only the operations of addition, subtraction, multiplication, and constant positive whole number exponents. A term is made up of coefficient and exponent.

Examples –
❑ Polynomial with single variable P(X) = **4**X^3 + **6**X^2+**7**X+9 (**4,6,7** are coefficient & **3,2** are exponent)
❑ Polynomial with 2 variables P(X, Y) = X^3 - **2**XY +1 (**2** is coefficient & **3** is exponent)

Definition–
❑ Polynomial with single variable P(X) = $\sum_{k=0}^{n} a_k x^k = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$

Polynomial can be represented using Linked List in 2 ways namely:
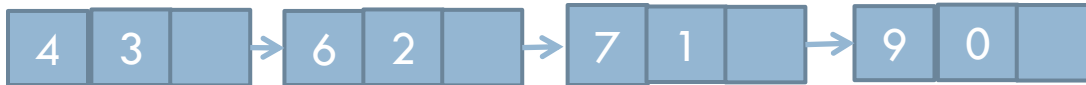1. Single Linked List
2. Header linked list

Single Linked List Polynomial Representation

Node of a polynomial:

Coefficient	Exponent	Next

For example, $P(X) = 4X^3 + 6X^2 + 7X + 9$ is represented as

4	3		→	6	2		→	7	1		→	9	0	

In each node the exponent field will store the corresponding exponent and the coefficient field will store the corresponding coefficient. Link field points to the next item in the polynomial.

Points to keep in mind

1. The sign of each coefficient and exponent is stored within the coefficient and the exponent itself
2. The storage allocation for each term in the polynomial must be done in descending order of their exponent

School of Computer Engineering

Header Linked List Polynomial Representation

Let p(x) denote the following polynomial in one variable (containing four nonzero terms): $P(x) = 2x^8 - 5x^7 - 3x^2 + 4$. It may be represented by the header list shown below where each node corresponds to a nonzero term of p(x). Specifically, the information part of the node is divided into two fields representing the coefficient and the exponent of the corresponding term and the nodes are linked according to decreasing degree of the exponent. The header node is needed to represent the zero polynomial (i.e. a polynomial with the value zero (0) is known as zero polynomial)
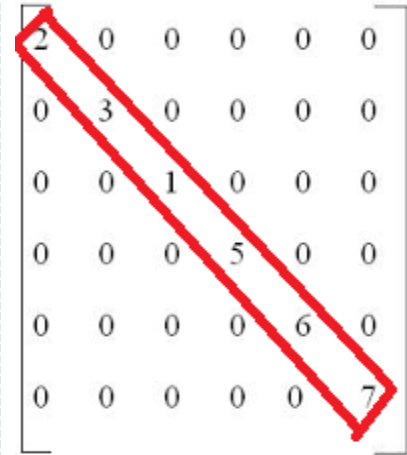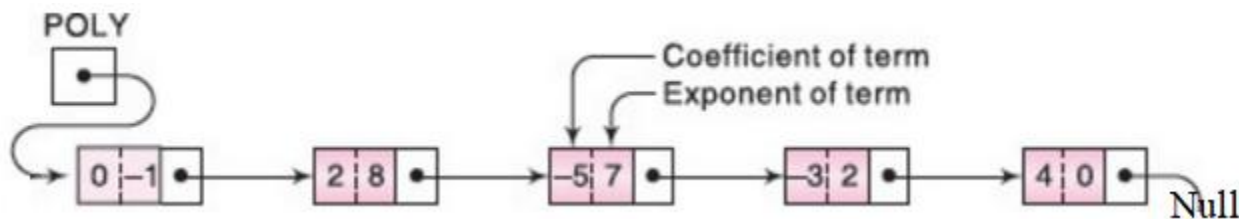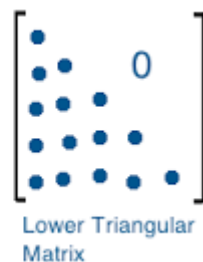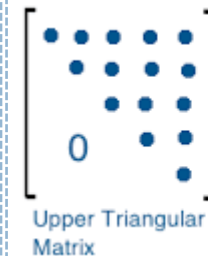
Sparse Matrix

A **sparse matrix** is a two-dimensional array in which most of the elements are zero or null. It is a wastage of memory and processing time if we store null values or 0 of the matrix in an array. To avoid such circumstances different techniques are used such as linked list.

N-square sparse matrices is called **triangular matrix**. A square matrix is called **lower triangular** if all the entries above the main diagonal are zero. Similarly, a square matrix is called **upper triangular** if all the entries below the main diagonal are zero.

Upper Triangular Matrix

Lower Triangular Matrix

Sparse Matrix cont…

Sparse matrix can be represented using a list in 2 ways –

Single Linked List and *Header linked list*

1. **Single Linked List:** For example, consider the sparse matrix

0	2	0	0	2	0
0	0	0	1	0	5
0	0	4	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Node Structure →

```
struct node
{
    int row;
    int column;
    int value;
    struct node *next;
} *start;
```

The linked list can be represented using linked list as shown below

start → | 0 | 1 | 2 | → | 0 | 4 | 2 | → | 1 | 3 | 1 | → | 1 | 5 | 5 | → | 2 | 2 | 4 | → nil

Class Work

Design non-array based data structure to represent lower triangular, upper triangular and tri-diagonal matrix.

School of Computer Engineering
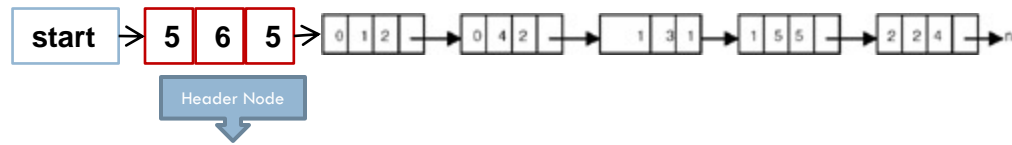
Sparse Matrix cont...

Header Linked List: For example, consider the sparse matrix

0	2	0	0	2	0
0	0	0	1	0	5
0	0	4	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Node Structure →

```
struct node
{
    int row;
    int column;
    int value;
    struct node *next;
} *start;
```

The linked list can be represented using linked list as shown below

start → | 5 | 6 | 5 | → | 0 1 2 | → | 0 4 2 | → | 1 3 1 | → | 1 5 5 | → | 2 2 4 | → nil

Header Node

Represents the header node and consists of:

1. Number of rows in the sparse matrix
2. Number of columns in the sparse matrix
3. Number of non-zero elements in the sparse matrix

School of Computer Engineering

Assignments

1. Design algorithm/develop pseudocode/write C code to add a given value K to each element in the LIST.
2. Design algorithm/develop pseudocode/write C code to deletes the last node from the LIST.
3. Design algorithm/develop pseudocode/write C code to delete the 1st node from the list.
4. Design algorithm/develop pseudocode/write C code which interchanges the Kth and K+1st elements
5. Design algorithm/develop pseudocode/write C code to swap nodes in a linked list without swapping data.
6. Design algorithm/develop pseudocode/write C code to reverse the nodes in a linked list.
7. Design algorithm/develop pseudocode/write C code to create a linked list from a given linked list. The new linked list must contain every alternate element of the existing linked list.
8. Design algorithm/develop pseudocode/write C code to count the number of times a given key occurs in a linked list
9. Let a linked list consists of n number of nodes, where each node consists of an unique number, a priority number(in between 1 to 5), and pointer to next node Design the algorithm/develop pseudocode/write C code to divide the nodes into different linked list where each linked consists of nodes having same priority.

School of Computer Engineering

Assignments

10. Design algorithm/develop pseudocode/write C code to delete n nodes after m nodes of a linked list

11. Design algorithm/develop pseudocode/write C code to check if a singly linked list is palindrome

12. Design algorithm/develop pseudocode/write C code to search an element in a linked list, if found delete that node and insert that node at beginning. Otherwise display an appropriate message.

13. Design algorithm/develop pseudocode/write C code to add, subtract and multiply 2 polynomials.

14. Design algorithm/develop pseudocode/write C code to delete last occurrence of an item from linked list

15. Given a singly linked list with nodes L_0 -> L_1 -> ... -> L_{n-1} -> L_n. Design the algorithm/develop pseudocode/write C code to rearrange the nodes in the list so that the new formed list is : L_0 -> L_n -> L_1 -> L_{n-1} -> L_2 -> L_{n-2} ...

School of Computer Engineering

Home Work (HW)

1. Design algorithm/develop pseudocode/write C code to delete the whole linked List
2. Design algorithm/develop pseudocode/write C code to find kth node from the end of a linked list
3. Design algorithm/develop pseudocode/write C code to merge two sorted linked lists and create a sorted linked list.
4. Design algorithm/develop pseudocode/write C code to delete alternate nodes of a linked list.
5. Design algorithm/develop pseudocode/write C code to find the union of two linked list
6. Design algorithm/develop pseudocode/write C code to find the difference of two linked list
7. Design algorithm/develop pseudocode/write C code to find the intersection of two linked list
8. Design algorithm/develop pseudocode/write C code to delete all duplicate nodes from a doubly circular linked list.
9. Design algorithm/develop pseudocode/write C code to delete consecutive two nodes from a doubly linked list based on the given position
10. Design algorithm/develop pseudocode/write C code to add two sparse matrices,

Supplementary Reading

- http://www.geeksforgeeks.org/data-structures/linked-list/
- https://www.tutorialspoint.com/data_structures_algorithms/linked_list_algorithms.htm
- http://www.studytonight.com/data-structures/introduction-to-linked-list
- http://nptel.ac.in/courses/106103069/8
- https://www.tutorialspoint.com/learn_c_by_examples/linked_list_programs_in_c.htm

FAQ

❑ **What is Linked List?**

a linked list is a linear collection of data elements, in which linear order is not given by their physical placement in memory. Each pointing to next node by means of a pointer. It is a data structure consisting of a group of nodes which together represent a sequence. Under the simplest form, each node is composed of data and a reference (in other words, a link) to the next node in the sequence. This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration. More complex variants add additional links, allowing efficient insertion or removal from arbitrary element references.

FAQ
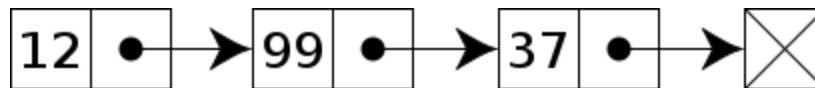
❑ **Linked List ADT**

ADT consists of the following primitive operations:
Create()
Destroy()
Add(element)
Remove(element)
Traverse()
IsEmpty()
Search(element)

❑ **Remove loop in Linked List**

Write a function that checks whether a given Linked List contains loop and if loop is present then removes the loop and returns true. And if the list doesn't contain loop then returns false. Above diagram shows a linked list with a loop. On passing it to the function, it should modify the below list to 1->2->3->4->5->NULL.