Hope Foundation's
International Institute of Information Technology,
Hinjawadi, Pune-57

Faculty Orientation Workshop on Data Structures

SE E&TC/ Electronics 2019 Course

Unit 4: LINKED LIST

Dr. Varsha Degaonkar
Department of Electronics and Telecommunication

Outline

Unit IV: Linked List (06 Hrs)

- Concept of linked organization,
- Singly Linked List,
- Stack using linked list,
- Queue using linked list,
- Doubly Linked List,
- Circular Linked List,
- Linked list as ADT.
- Representation & manipulations of polynomials using linked list,
- comparison of sequential and linked organization.

Mapping of Course Outcomes: CO4- Demonstrate applicability of Linked List.

CO-PO Mapping with Justification

Course Outcome	Blooms Taxonomy Level	After successful completion of the course students will be able to	Mapping with Syllabus Unit	PO MAPPING
CO204184.4	3	Demonstrate applicability of Linked List.	4	3, 4, 5, 12

MAPPING	LEVEL	JUSTIFICATION
CO4-PO3 (Design/development of solutions)	2	Design solutions for complex engineering problems using Linked List.
CO4-PO4 (Conduct investigations of complex problems)	1	Design of experiments, interpretation of data, and synthesis of the information using dynamic memory allocation to provide valid conclusions.
CO4-PO5 (Modern tool usage)	1	Select appropriate IT tools for modeling linear data structures using dynamic memory allocation.
CO4-PO12 (Life-long learning)	1	Recognize the need for dynamic memory allocation, and have the preparation for technological change.

Content – Book – Pages Mapping

	UNIT 4: Linked List	
	• Concept of linked organization, • Singly linked list, • Stack using linked list, queue using linked list, • Doubly linked list, • Circular linked list, • Linked list as ADT.	T1 (134-137) T1 (138-146, 162-163) T1 (146-150) T1 (164,165, 179-182) T1 (186-188) T1(137)
	• Representation and manipulations of polynomials using linked lists, • Comparison of sequential organization with linked organization	T1 (150-151) T1 (145-146)

Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Content coverage and depth

4.1 Concept of linked organization:

- **List**
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

List

List: a set of items organised sequentially

Ex. Array

Disadvantages of array:

1) The size of the arrays is fixed

2) Inserting a new element in an array of elements is expensive because the space has to be created for the new elements and to create space existing elements have to be shifted.

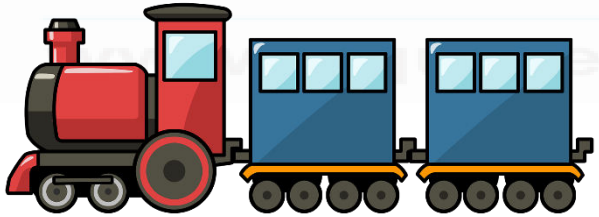
Content coverage and depth

4.1 Concept of linked organization:

- List
- **Linked List (LL)**
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Linked List

- A linked list is a **linear data structure.**
- All the elements **are not stored at contiguous memory locations.**
- It's a list where the **elements are linked** using pointers.
- Linked List Data Structure is **Dynamic** in nature.
- **Memory will be allocated at run time** i.e while running program.



Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- **Memory Allocation: Static & Dynamic**
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Memory Allocation: Static & Dynamic

https://www.youtube.com/watch?v=SuBch2MZpZM

Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- **Comparison of sequential and linked organization**
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Comparison of sequential and linked organization

S. No	Sequential organization	Linked organization
1.	Insertions and deletions are difficult.	Insertions and deletions can be done easily.
2.	It needs movements of elements for insertion and deletion.	It does not need movement of elements for insertion and deletion.
3.	In it space is wasted.	In it space is not wasted.
4.	It is more expensive.	It is less expensive.
5.	It requires less space as only information is stored.	It requires more space as pointers are also stored along with information.
6.	Its size is fixed.	Its size is not fixed.
7.	It can not be extended or reduced according to requirements.	It can be extended or reduced according to requirements.
8.	Same amount of time is required to access each element.	Different amount of time is required to access each element.
9.	Elements are stored in consecutive memory locations.	Elements may or may not be stored in consecutive memory locations.
10.	If have to go to a particular element then we can reach there directly.	If we have to go to a particular element then we have to go through all those elements that come before that element.

http://www.xpode.com/ShowArticle.aspx?Articleid=282

Content coverage and depth

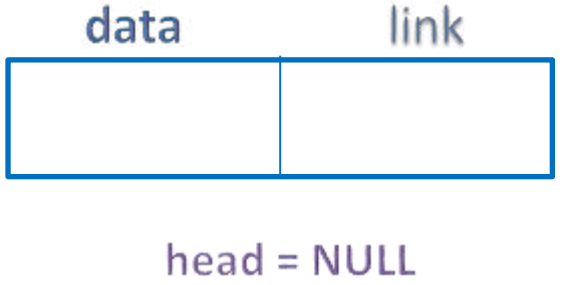
4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- **Representation of node in LL**
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Representation of node in LL

- A linked list consists of nodes where each node contains data fields and reference (link) to the next node in the list.
- The first node is called the head.
- If the linked list is empty, then the value of the head is NULL.
- **Each node in a list consists of at least two parts**:

  1) Data 2) Pointer (or Reference) to the next node



Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- **Structure of LL**
- Self-referential structure
- Advantages of LL
- Types of LL
- Linked list as ADT

Structure of Linked List

```
struct node
  { int data;
    struct node *next;
  };
void main()
{
 -----
 struct node *head=NULL;
-----
}
```

```
typedef struct node
  { int data;
    struct node *next;
  }node;
void main()
{
 -------
 node *head=NULL;
 -----
}
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- **Self-referential structure**
- Advantages of LL
- Types of LL
- Linked list as ADT

Self-referential structure

```
struct node
  { int data;
    struct node *next;
  };
void main()
{
 -----
 struct node *head=NULL;
-----
}
```

structure contain member field that points to the same structure type.

Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- **Advantages of LL**
- Types of LL
- Linked list as ADT

Advantages of LL

Advantages of LL:

- Linked List is **Dynamic data Structure .**
- Linked List **can grow and shrink during run time.**
- **Insertion and Deletion** Operations are Easier
- **Efficient Memory Utilization**, i. e. no need to pre-allocate memory
- Faster Access time, can be expanded in **constant time without memory overhead**
- Linear Data Structures such as Stack, Queue can be **easily implemented** using Linked list

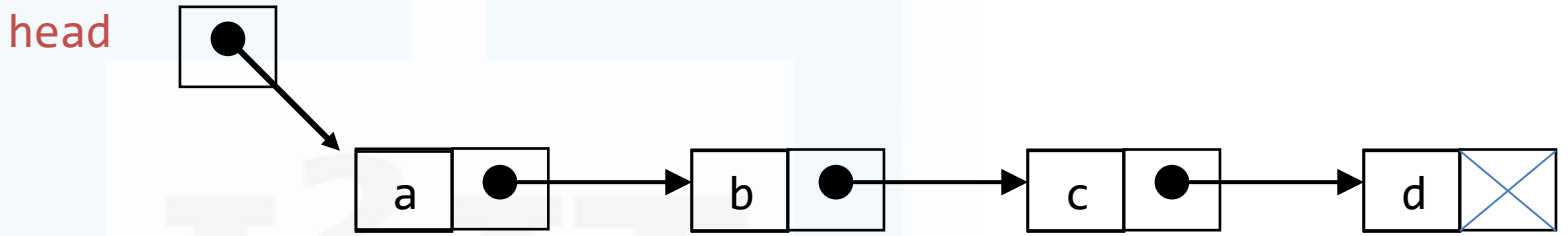
Content coverage and depth

4.1 Concept of linked organization:

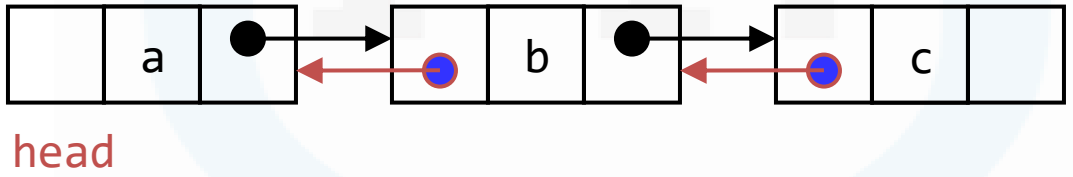
- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- **Types of LL**
- Linked list as ADT

Types of LL

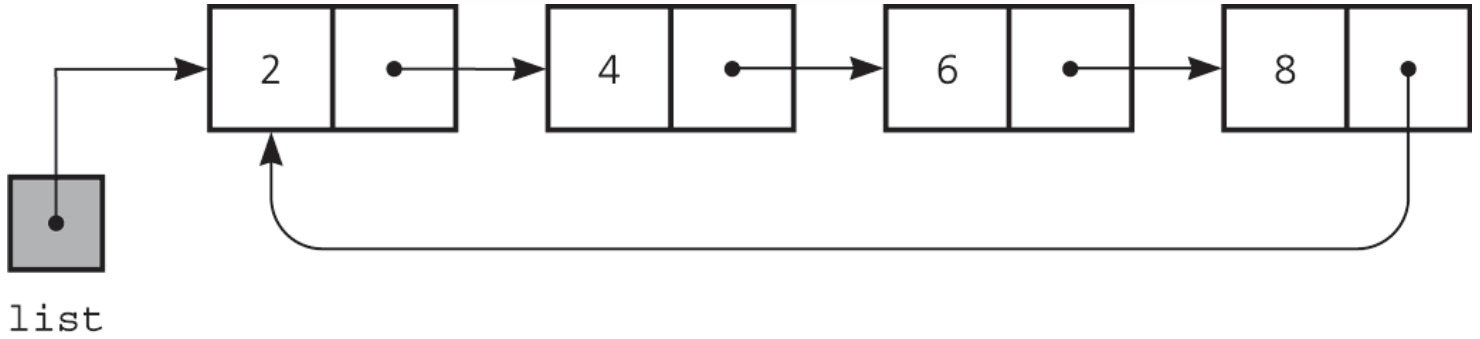
- Singly Linked List



- Doubly Linked List



- Circular Linked List



Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

Content coverage and depth

4.1 Concept of linked organization:

- List
- Linked List (LL)
- Memory Allocation: Static & Dynamic
- Comparison of sequential and linked organization
- Representation of node in LL
- Structure of LL
- Self-referential structure
- Advantages of LL
- Types of LL
- **Linked list as ADT**

Content coverage and depth

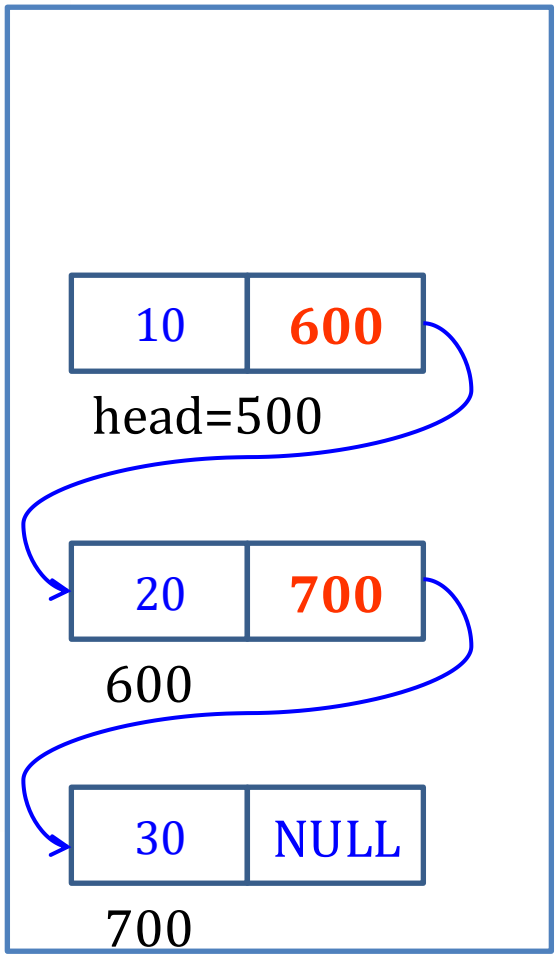
4.2 Singly Linked List (SLL):

1. Representation of SLL
2. Operations: Insertion, Deletion, Searching, Traversal

Representation of Singly Linked List (SLL):

Singly Linked List (SLL):

```
typedef struct sll
  { int data;
    struct sll *next;
  }sll;
```



Operations on Singly Linked List (SLL)

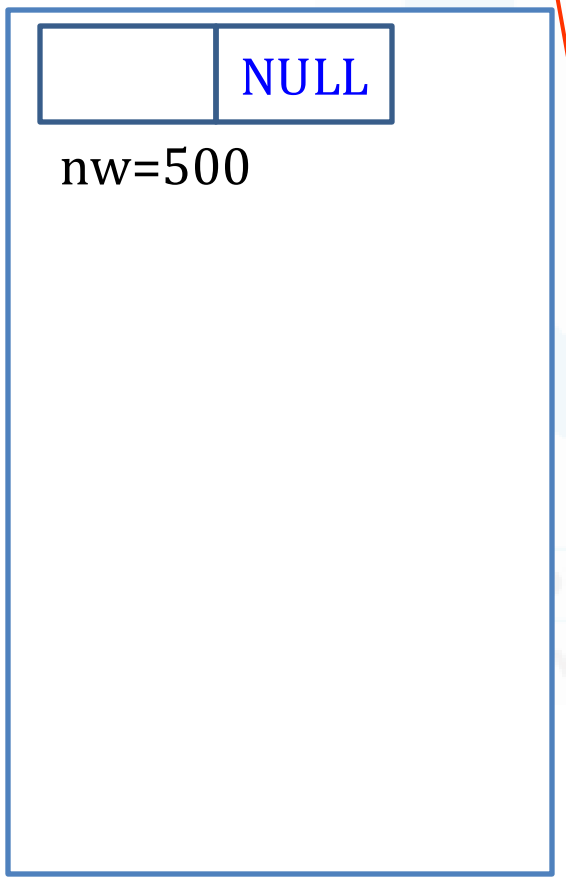
- Insertion
- Deletion
- Searching
- Traversal

Operations on Singly Linked List (SLL)

- Creation of Singly Linked List

```
head=create(head);      // head=NULL
/*Function Call*/
```

```
head=create(head);
/*Function Call*/
// head=NULL
```

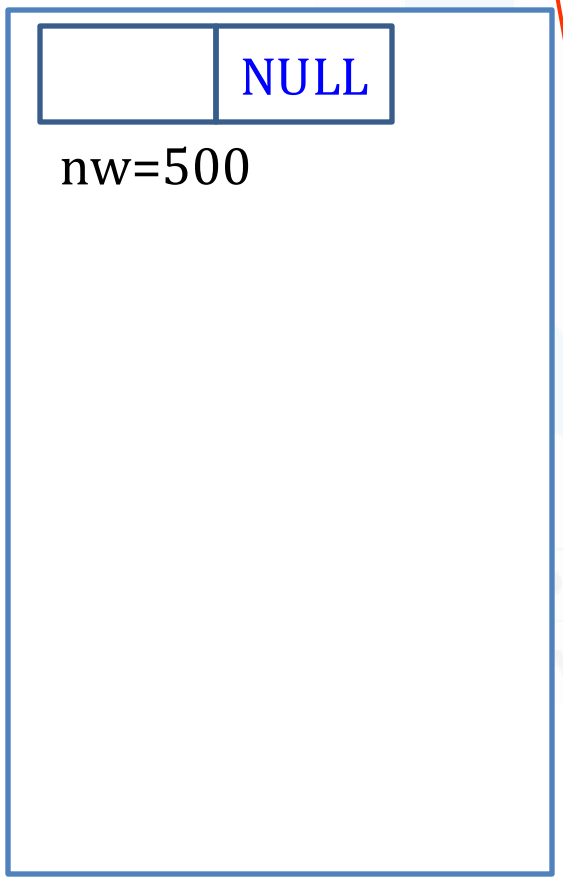



```
//create new node
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
```

Singly Linked List (SLL):

```
typedef struct sll
  { int data;
    struct sll *next;
  }sll;
```

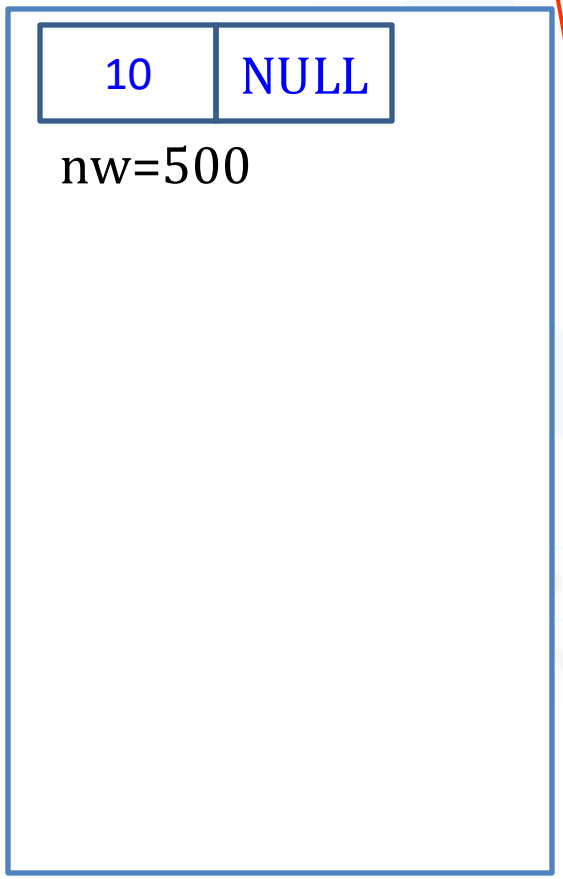
```
head=create(head);
/*Function Call*/
// head=NULL
```




```
//create new node
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

```
head=create(head);
/*Function Call*/
// head=NULL
```



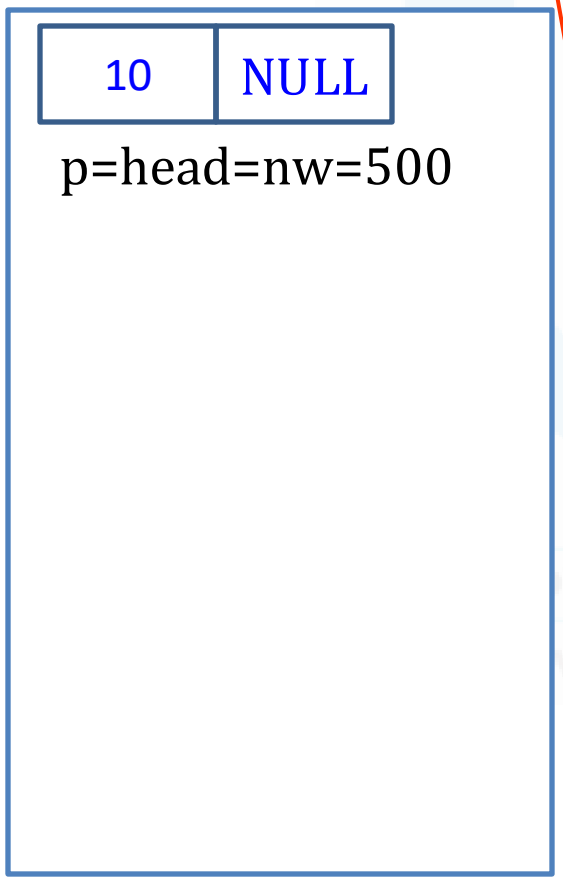

```
//create new node
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
```

```
head=create(head);
/*Function Call*/
// head=NULL
```

```
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
        p=head=nw;
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

```
head=create(head);
/*Function Call*/
```

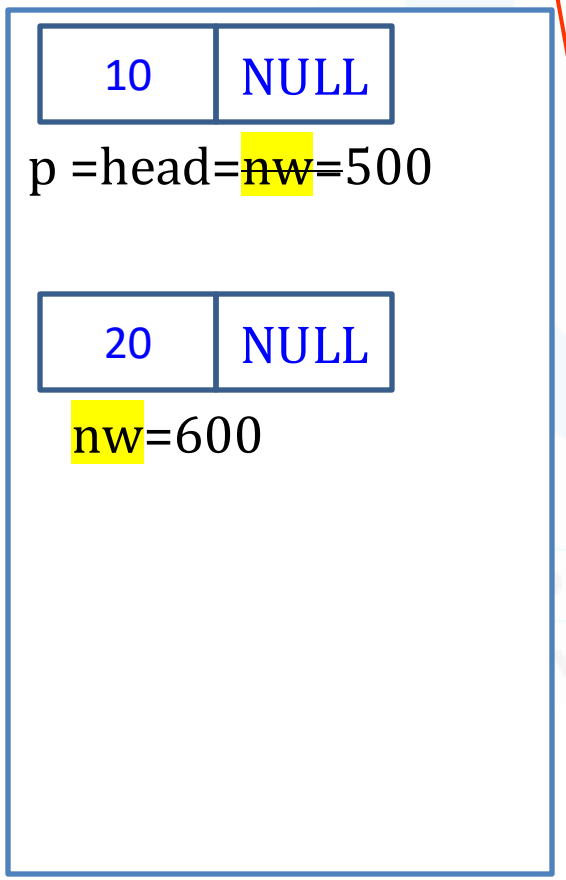



```
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
        p=head=nw;
```

```
head=create(head);
/*Function Call*/
```

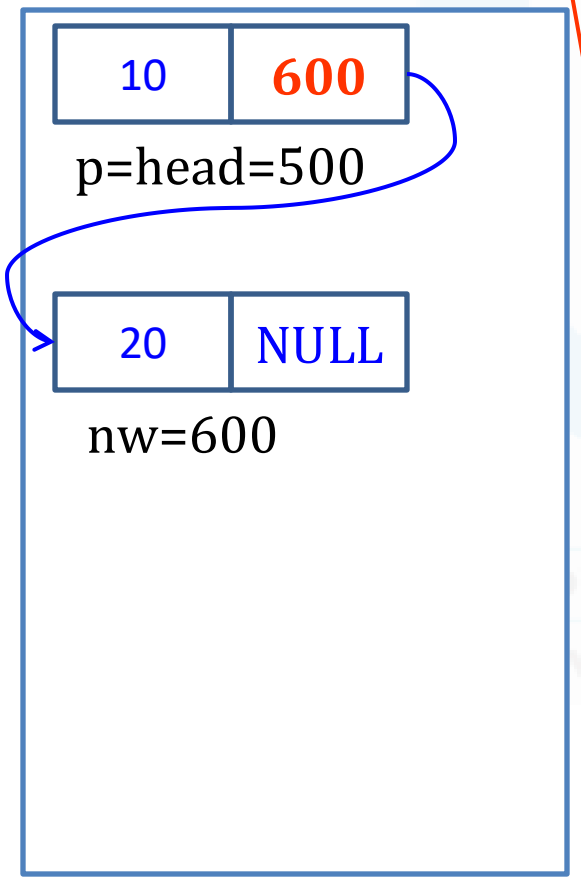
```
 do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
            p=head=nw;
 else {
        }
 printf("\n\t Do you want to insert node(Y/N)");
flushall();
 scanf("%c",&ans);
 }while(ans=='y'||ans=='Y');
```

```
head=create(head);
/*Function Call*/
```



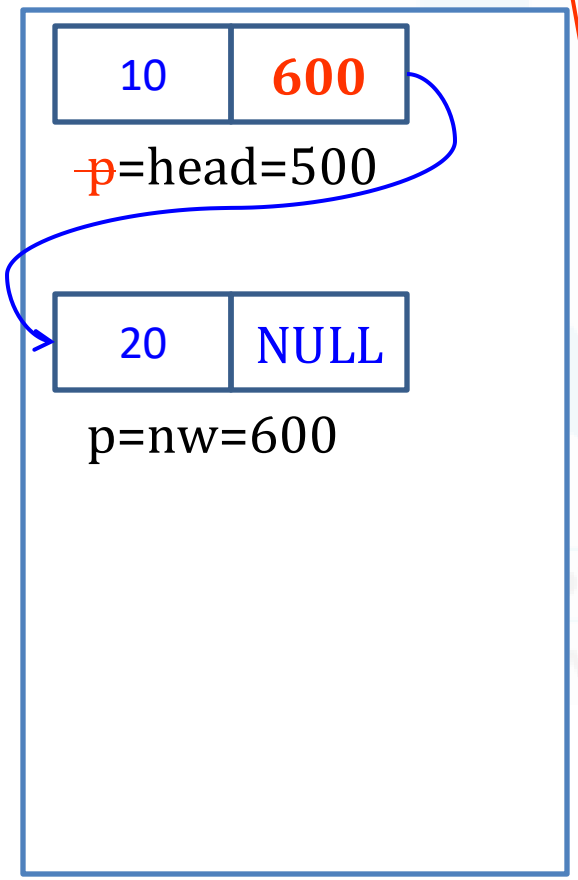

```
do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
if(head==NULL)
        p=head=nw;
else {
        p->next=nw;
        p=nw;
      }
printf("\n\t Do you want to insert node(Y/N)");
flushall();
scanf("%c",&ans);
}while(ans=='y'||ans=='Y');
```

```
head=create(head);
/*Function Call*/
```



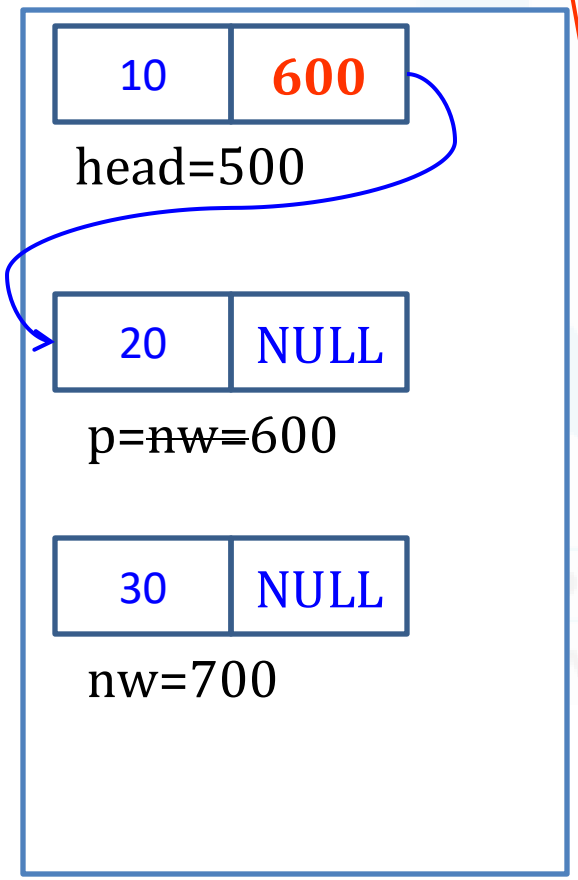

```
do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
            p=head=nw;
 else {    p->next=nw;
           p=nw;
        }
 printf("\n\t Do you want to insert node(Y/N)");
flushall();
 scanf("%c",&ans);
 }while(ans=='y'||ans=='Y');
```

```
head=create(head);
/*Function Call*/
```



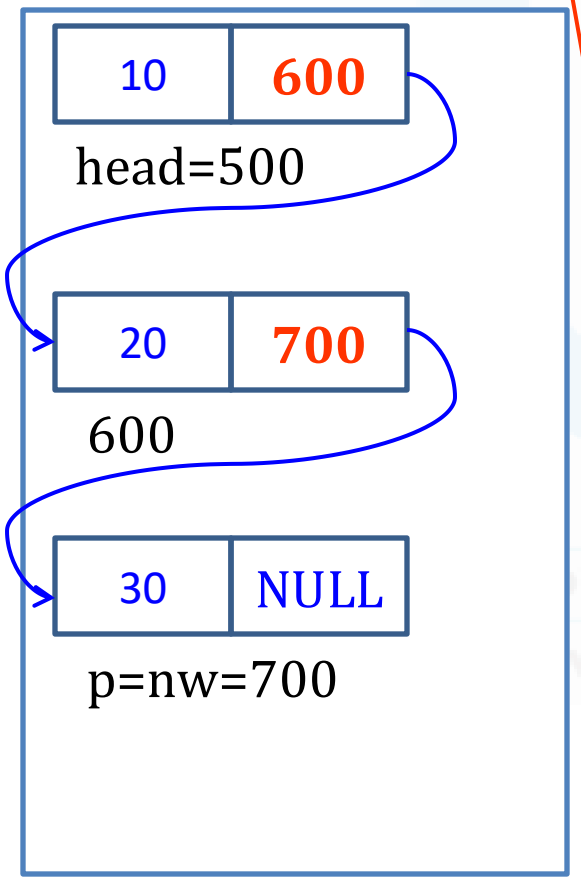

```
do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
          p=head=nw;
 else {   p->next=nw;
          p=nw;
       }
 printf("\n\t Do you want to insert node(Y/N)");
flushall();
 scanf("%c",&ans);
 }while(ans=='y'||ans=='Y');
```

```
head=create(head);
/*Function Call*/
```




```
do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
          p=head=nw;
 else {   p->next=nw;
          p=nw;
       }
 printf("\n\t Do you want to insert node(Y/N)");
flushall();
 scanf("%c",&ans);
 }while(ans=='y'||ans=='Y');
```

```
head=create(head);
/*Function Call*/
```

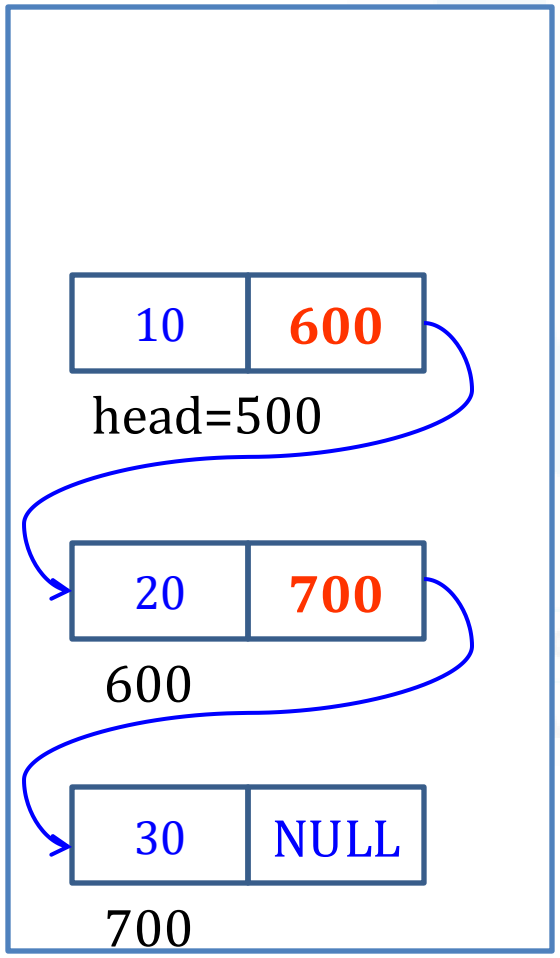



```
do{
nw=(sll*)malloc(sizeof(sll));
nw->next=NULL;
printf("\nEnter the data:");
scanf("%d",&(nw->data));
 if(head==NULL)
            p=head=nw;
 else {     p->next=nw;
            p=nw;
       }
 printf("\n\t Do you want to insert node(Y/N)");
flushall();
 scanf("%c",&ans);
 }while(ans=='y'||ans=='Y');
```

Operations on Singly Linked List (SLL)

- Insertion of a node in Singly Linked List

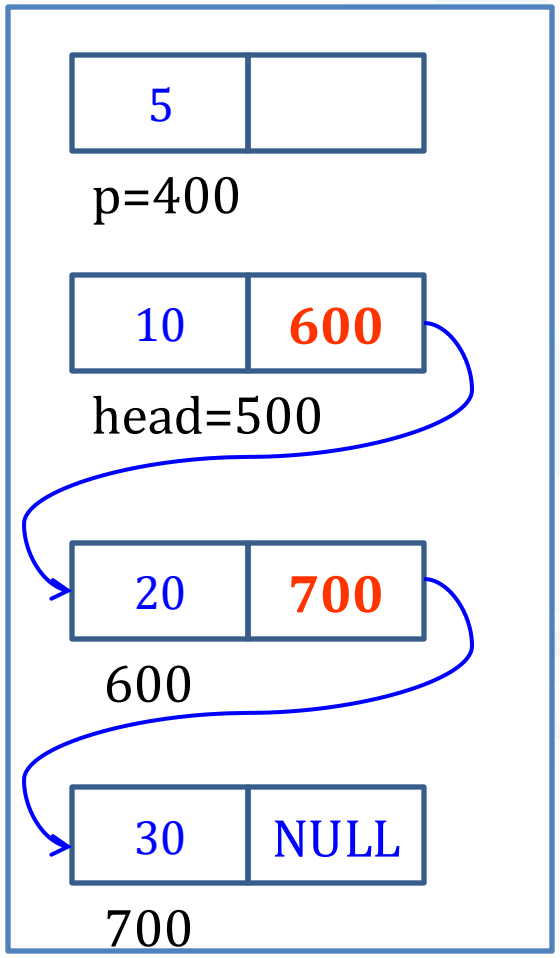
head= insert(head);




```
printf("\nEnter the location:");
scanf("%d",&loc);
```

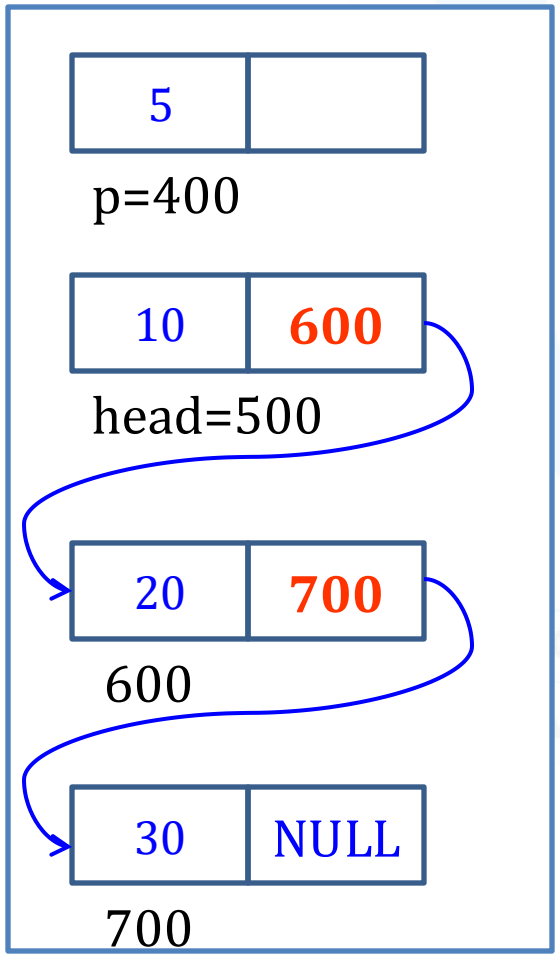

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= insert(head);




```
printf("\nEnter the location:");
scanf("%d",&loc);

p=(sll*)malloc(sizeof(sll));
printf("\nenter a data:");
scanf("%d",&(p->data));
```

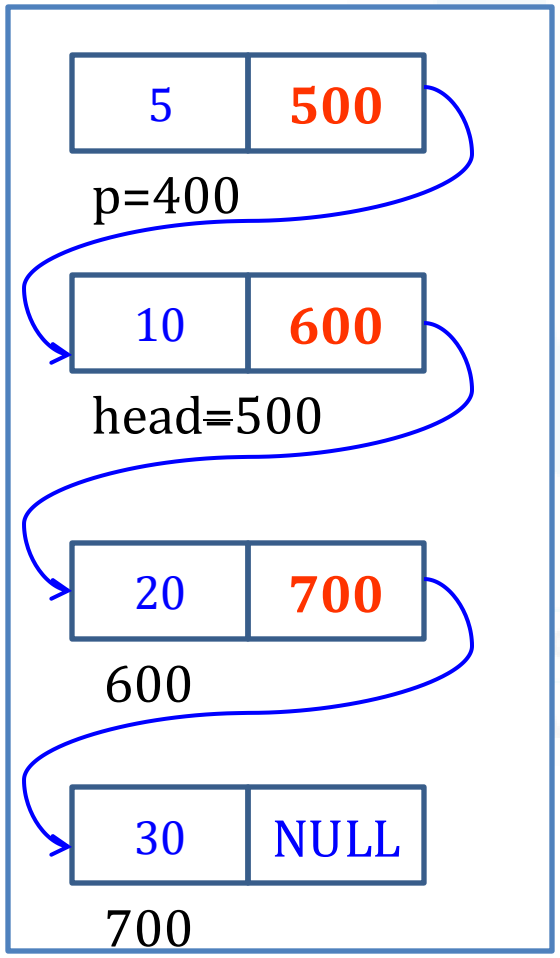
```
head= insert(head);

//loc=1
```




```
printf("\nEnter the location:");
scanf("%d",&loc);

p=(sll*)malloc(sizeof(sll));
printf("\nenter a data:");
scanf("%d",&(p->data));
```

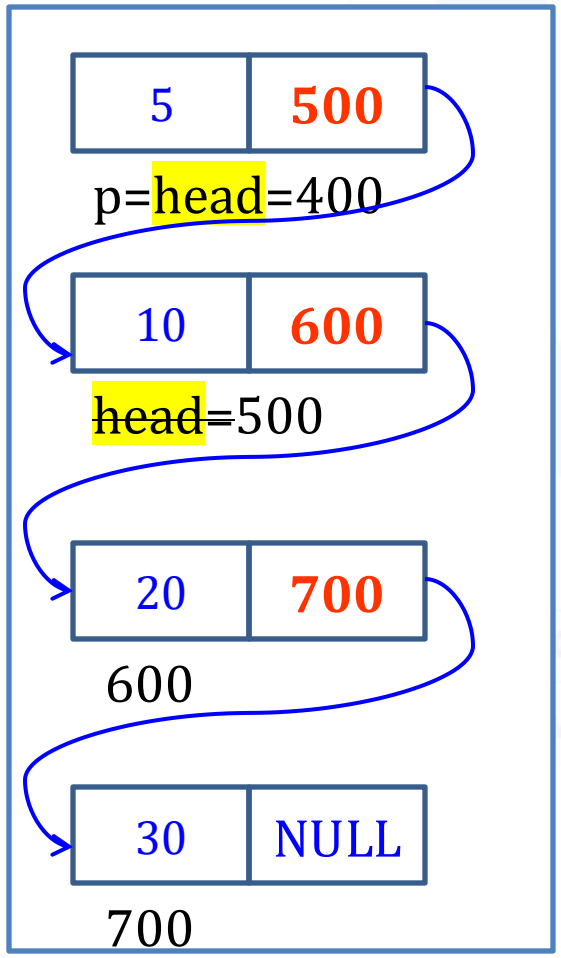

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= insert(head);



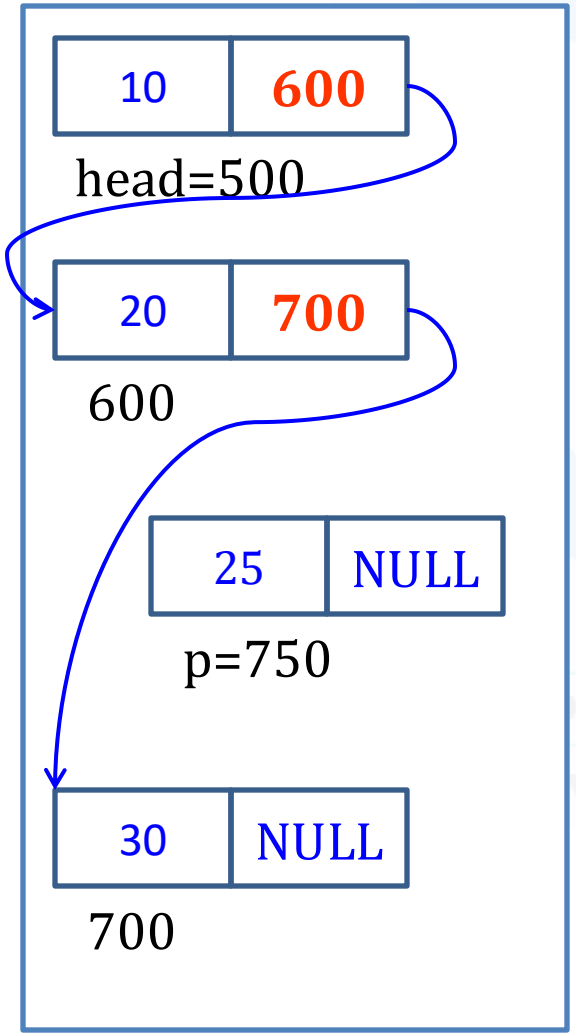

```
sll *insert(sll *head)
{ sll *p,*q;
  int loc,i;
  printf("\nEnter the location:");
  scanf("%d",&loc);
  p=(sll*)malloc(sizeof(sll));
  printf("\nenter a data:");
  scanf("%d",&(p->data));

  if(loc==1)
   { p->next=head;

   }
```

head= insert(head);



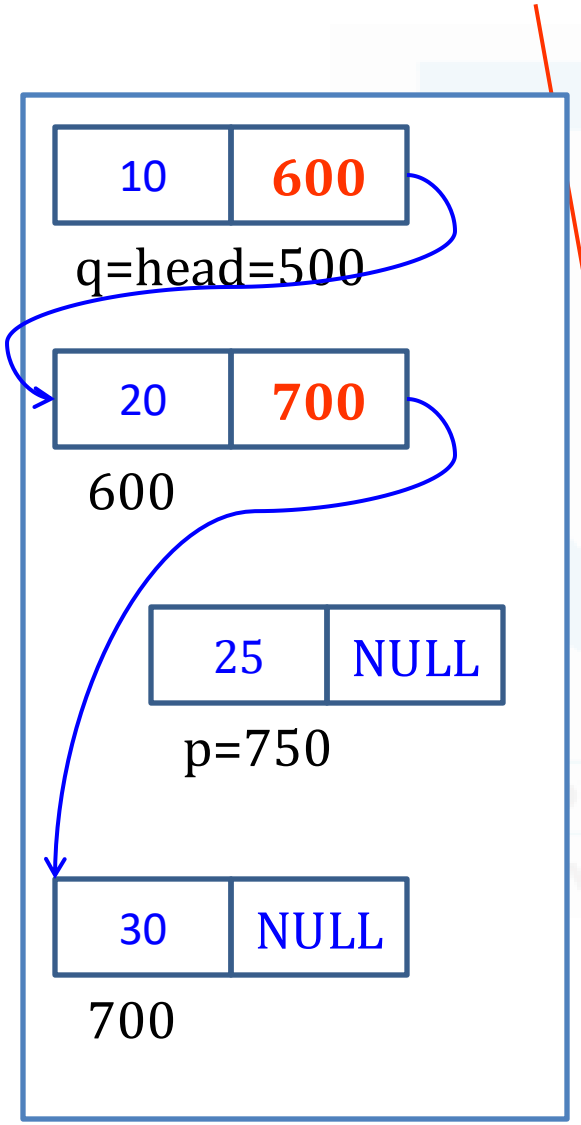

```
sll *insert(sll *head)
{ sll *p,*q;
  int loc,i;
  printf("\nEnter the location:");
  scanf("%d",&loc);
  p=(sll*)malloc(sizeof(sll));
  printf("\nenter a data:");
  scanf("%d",&(p->data));

  if(loc==1)
  { p->next=head;
    head=p;
    return(head);
  }
```

head= insert(head);



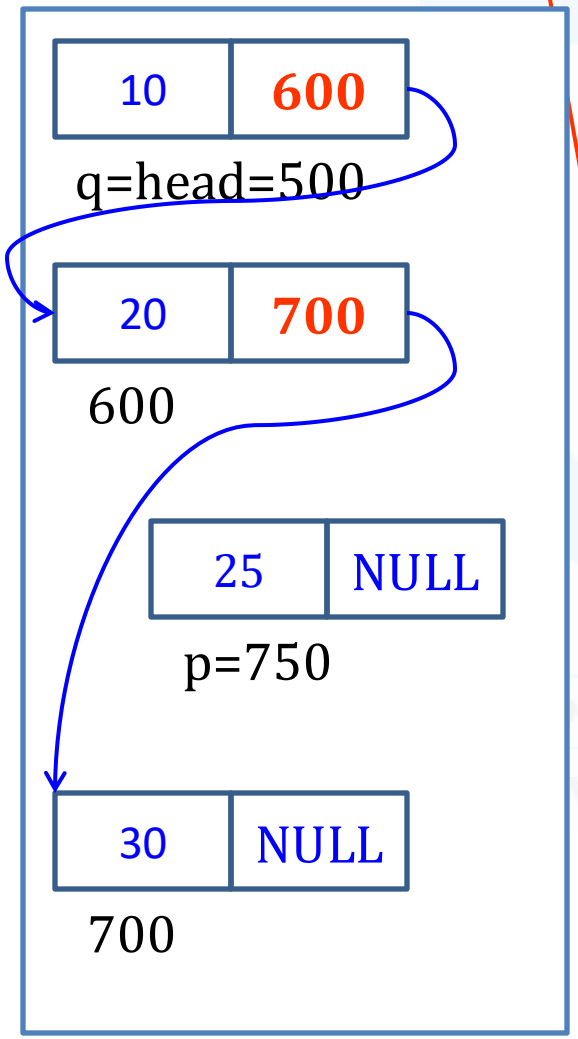
Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= insert(head);




```
else
  { i=1;q=head;
```

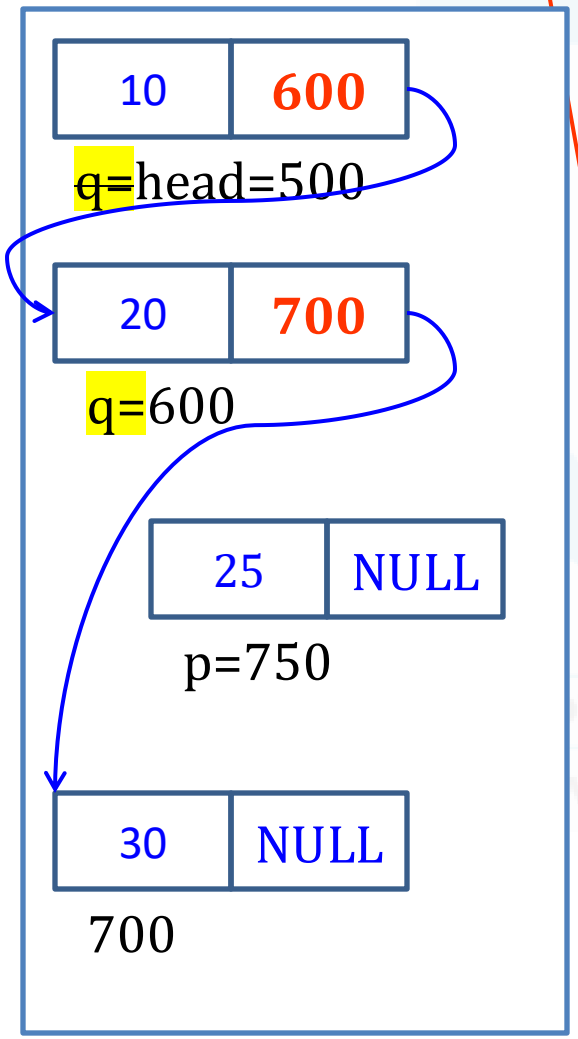
```
head= insert(head);
//loc=3
```




```
else
  { i=1;q=head;
   while(i<loc-1)
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

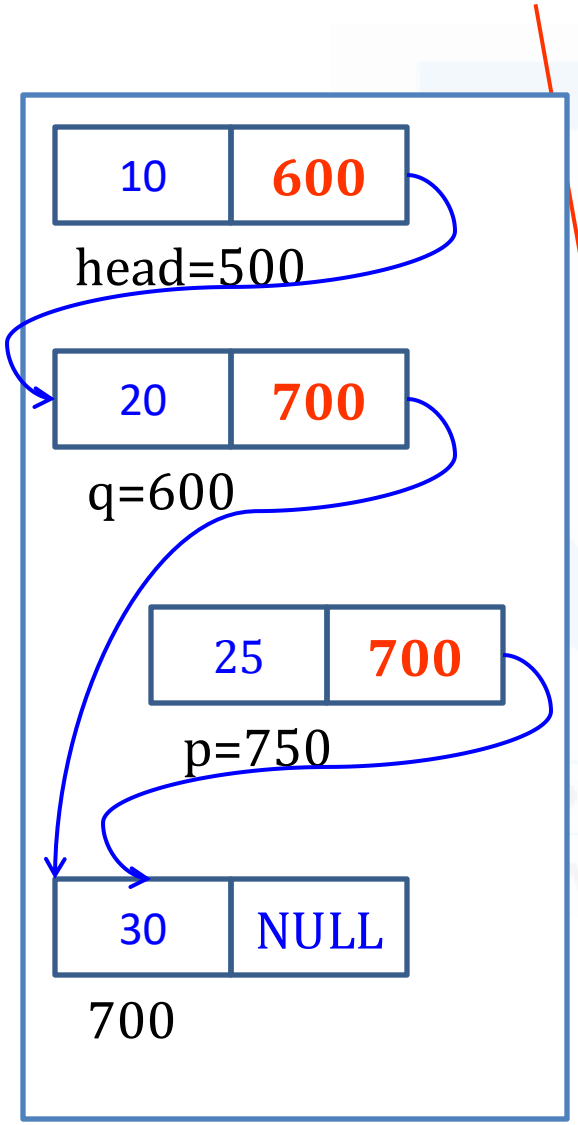
head= insert(head);




```
else
  { i=1;q=head;
   while(i<loc-1)
          {
                q=q->next;
                i++;
          }
```

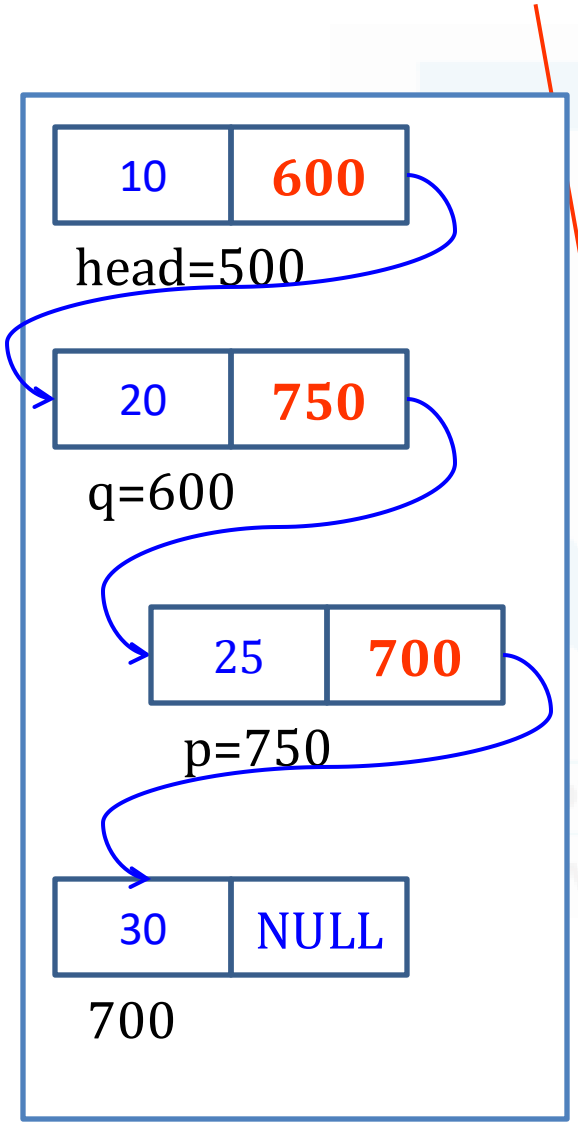

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= insert(head);




```
else
  { i=1;q=head;
   while(i<loc-1)
          {
                  q=q->next;
                  i++;
          }
  p->next=q->next;
  q->next=p;
```

head= insert(head);

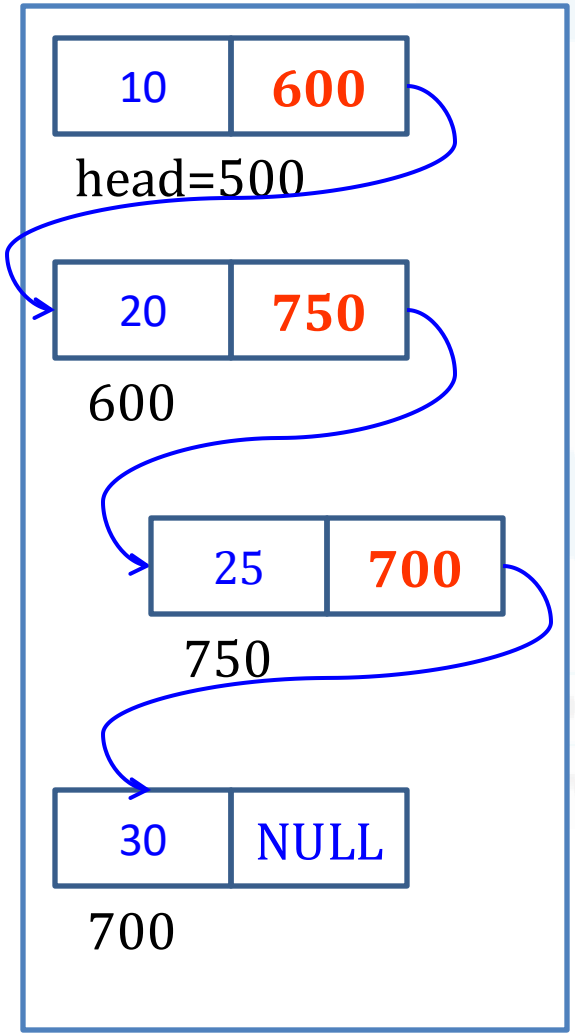



```
else
  { i=1;q=head;
   while(i<loc-1)
          {
                q=q->next;
                i++;
          }
  p->next=q->next;
  q->next=p;
  }
          disp(head);
```

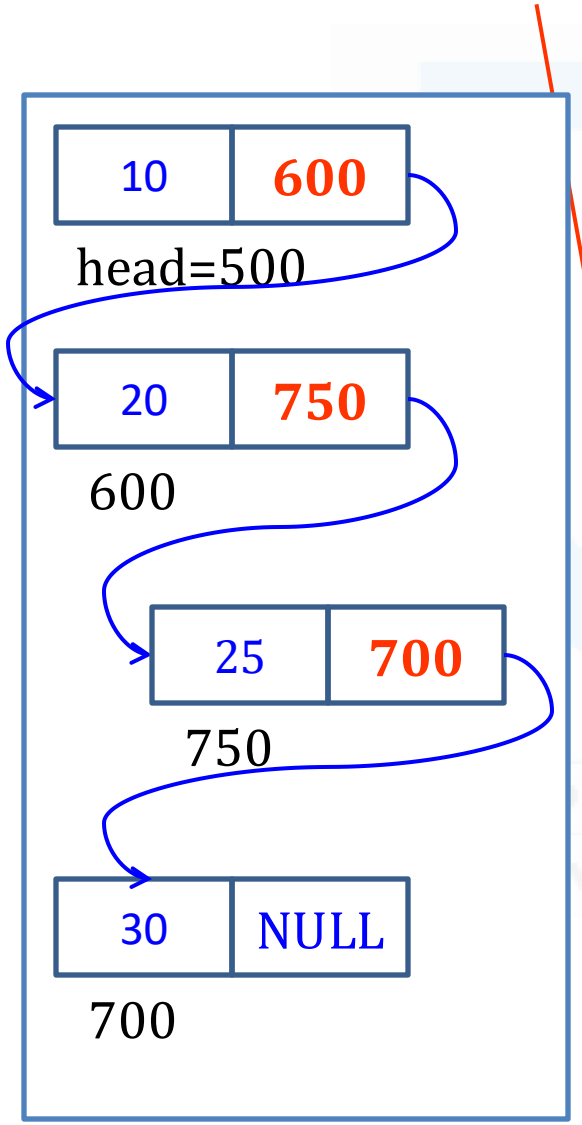
Operations on Singly Linked List (SLL)

- Deletion of a node from Singly Linked List

head= del(head);

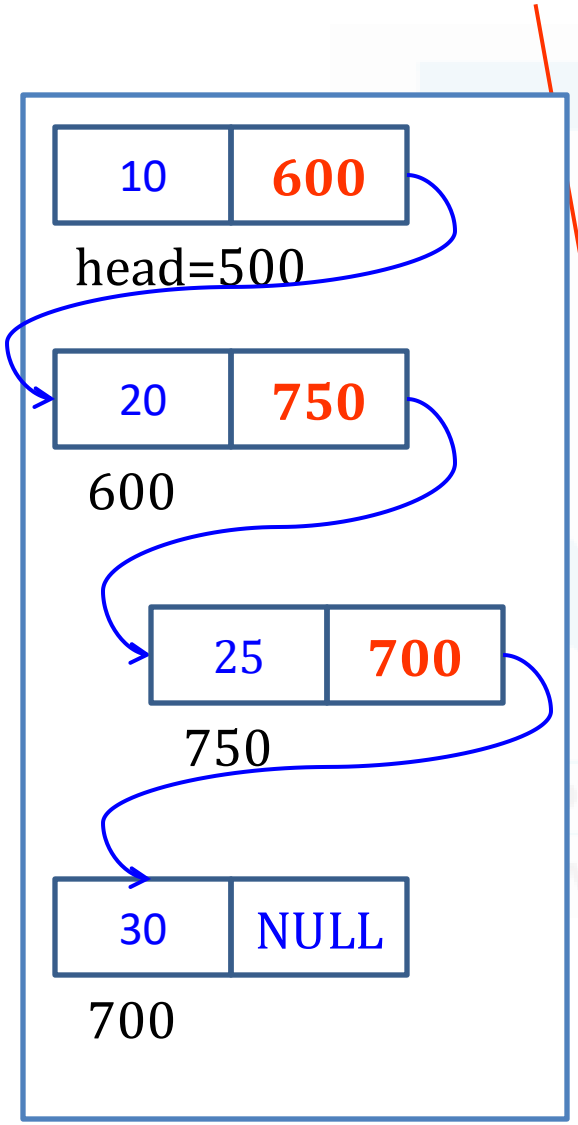


head= del(head);




```
if(head==NULL)
 {
  printf("\nEmpty Linked List");
  return(head);
 }
```

head= del(head);

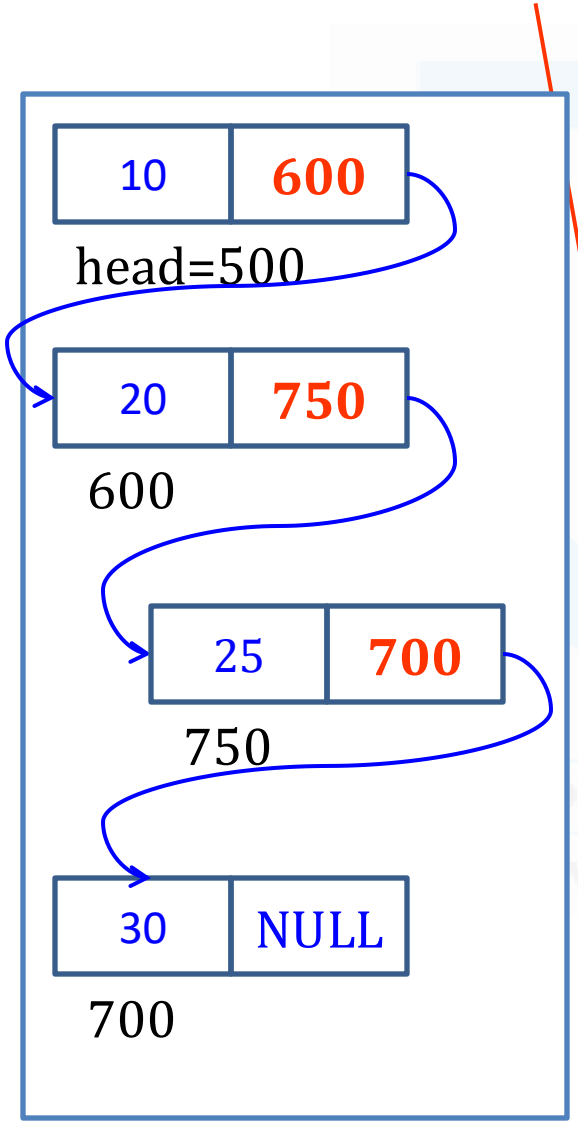



```
printf("\nEnter the location:");
scanf("%d",&loc);
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= del(head);

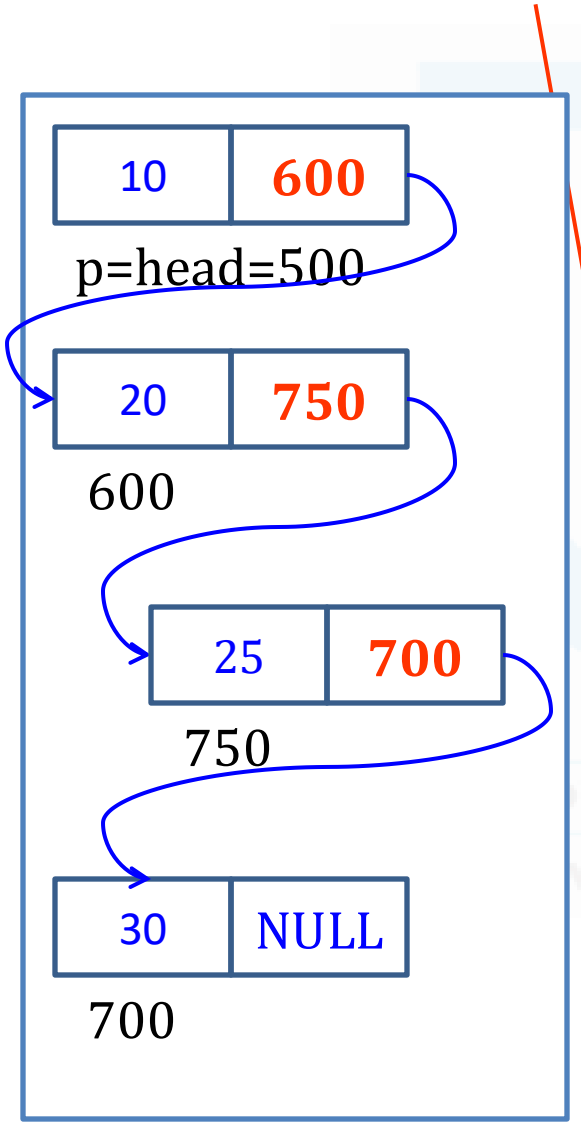
//loc=1




```
printf("\nEnter the location:");
scanf("%d",&loc);
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

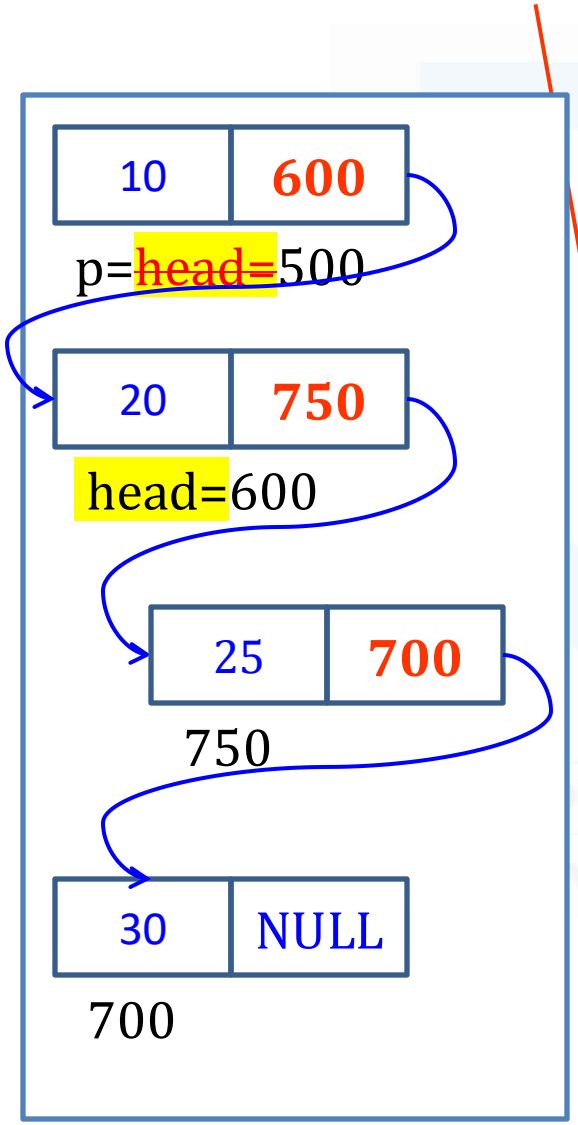
head= del(head);




```
if(loc==1)
 { p=head;



 }
```

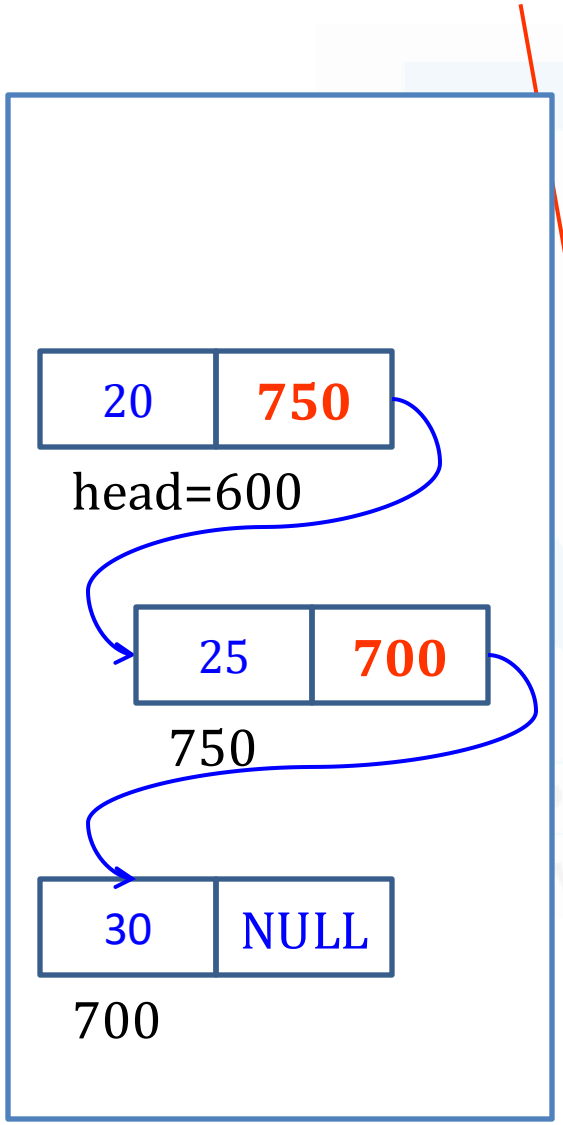

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= del(head);




```
if(loc==1)
 { p=head;
   head=head->next;
```

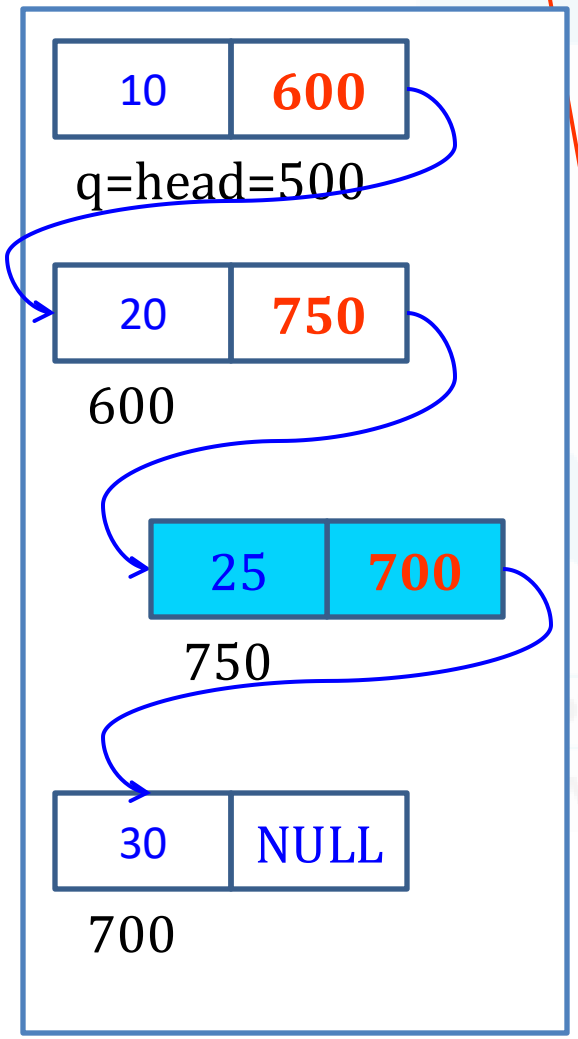
head= del(head);




```
if(loc==1)
 { p=head;
   head=head->next;
   free(p);
   printf("Data Deleted");
   return(head);
 }
```

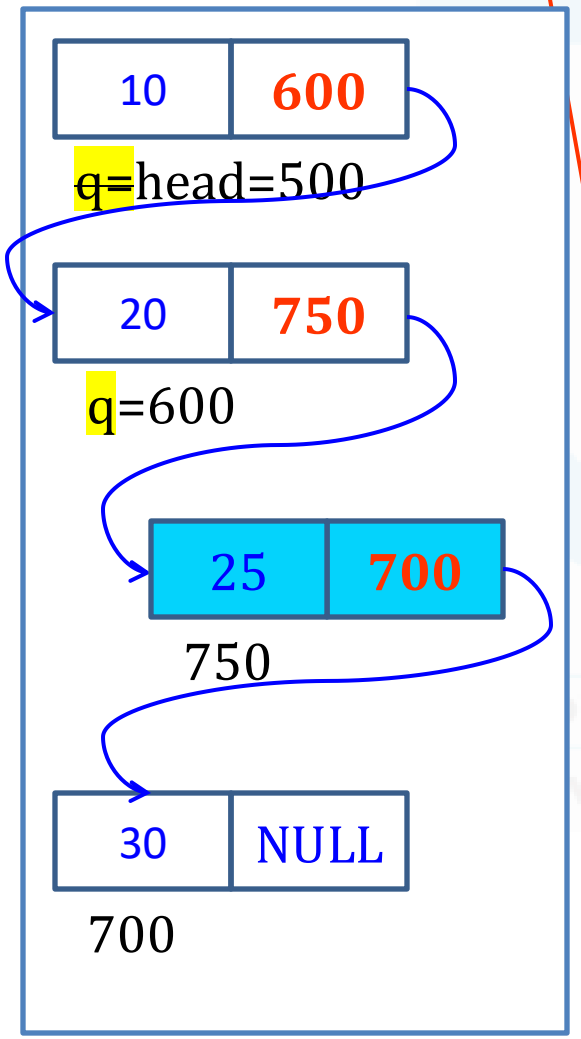
head= del(head);

//loc=3



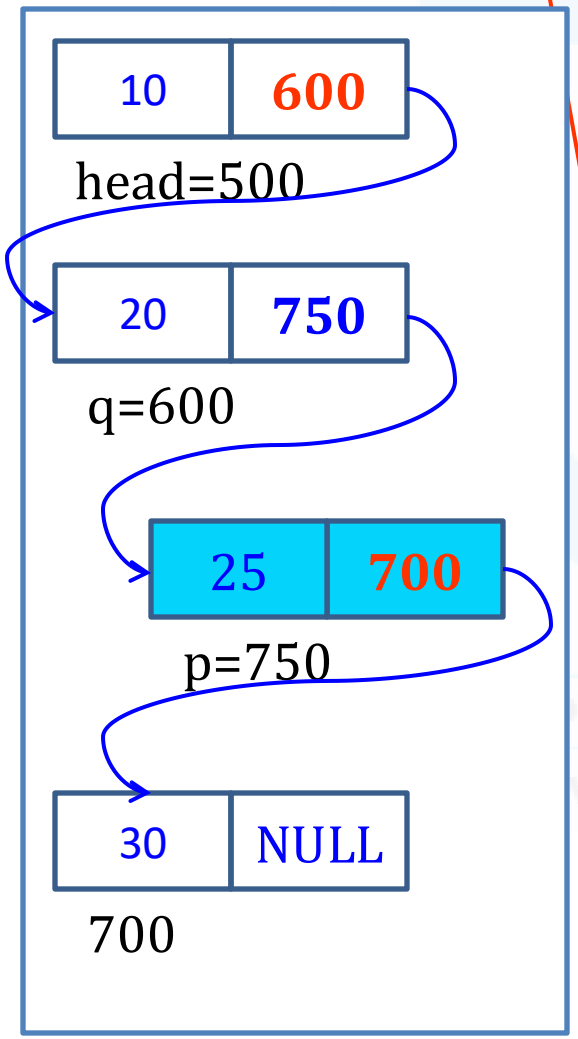

```
else
  {
   i=1;q=head;
```

head= del(head);



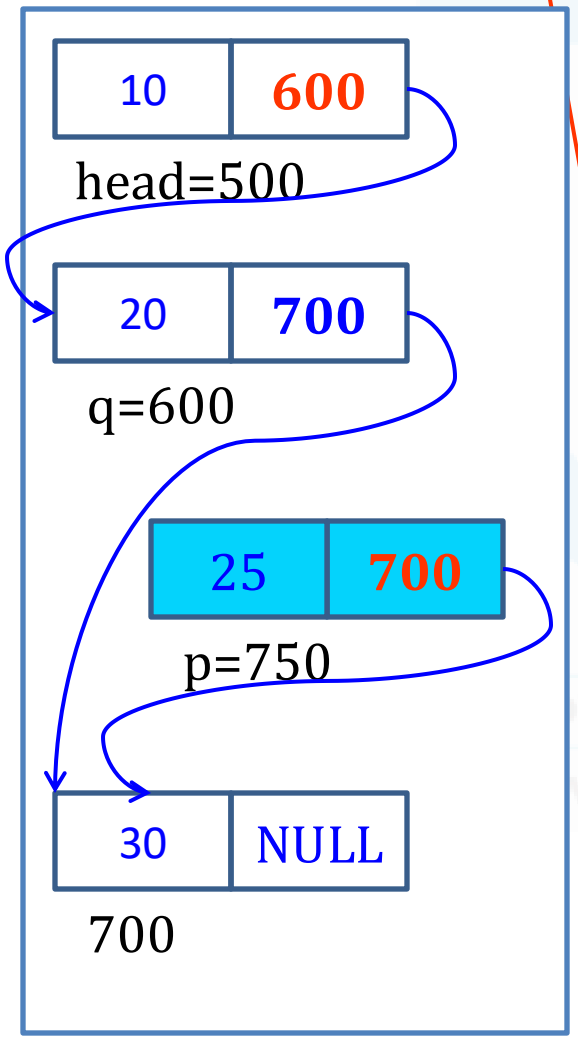

```
else
  {
   i=1;q=head;
   while(i<loc-1)
    {
          q=q->next;
          i++;
    }
```

head= del(head);



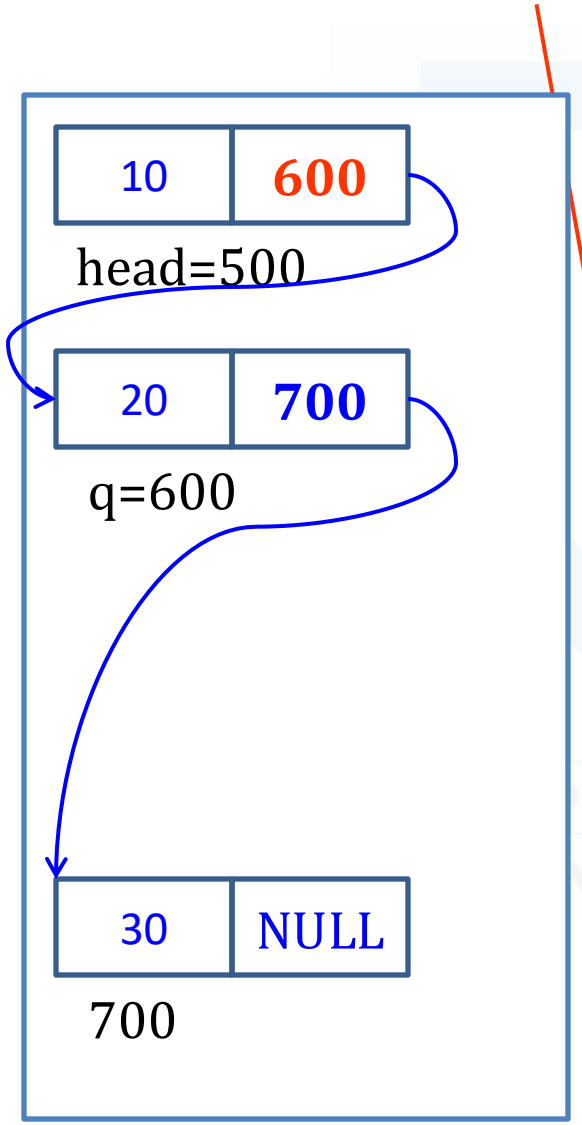

```
else
 {
  i=1;q=head;
  while(i<loc-1)
   {
        q=q->next;
        i++;
   }
 p=q->next;
 q->next=p->next;
 free(p);
```

head= del(head);




```
else
  {
   i=1;q=head;
   while(i<loc-1)
    {
          q=q->next;
          i++;
    }
  p=q->next;
  q->next=p->next;
  free(p);
  printf("Data Deleted");
          disp(head);
  return(head);
}
```

head= del(head);

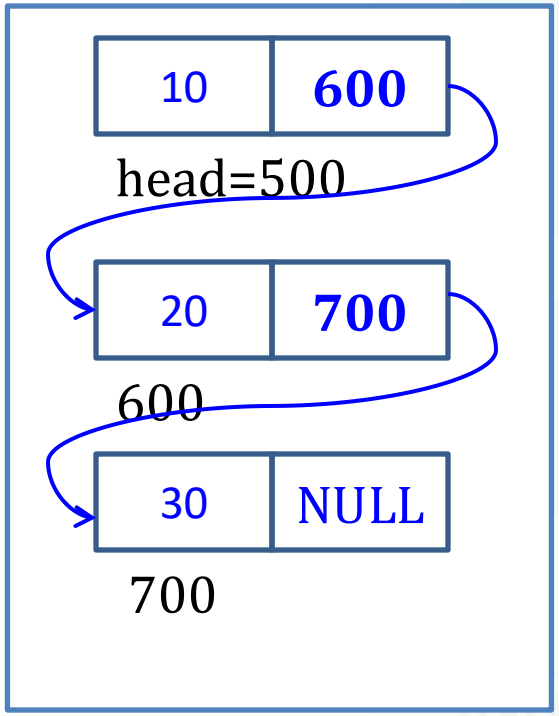



```
else
  {
   i=1;q=head;
   while(i<loc-1)
    {
          q=q->next;
          i++;
    }
  p=q->next;
  q->next=p->next;
  free(p);
  printf("Data Deleted");
          disp(head);
  return(head);
}
```

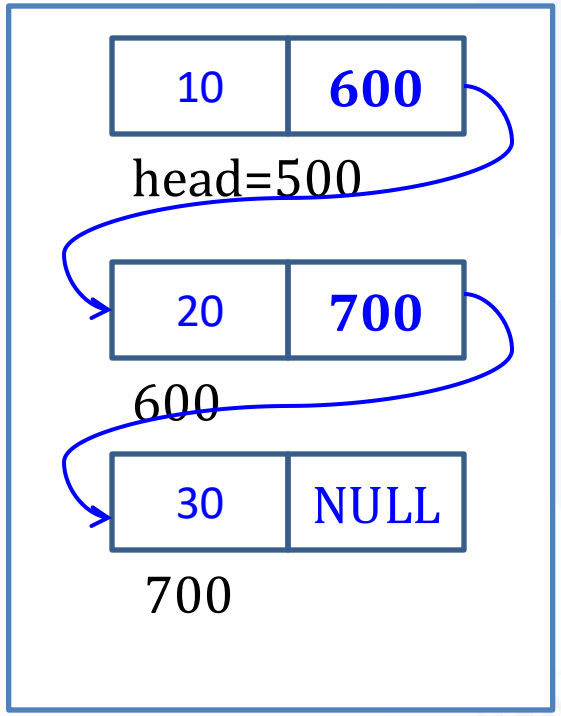
Operations on Singly Linked List (SLL)

- Searching of a node in Singly Linked List

head= search(head);



head= search(head);

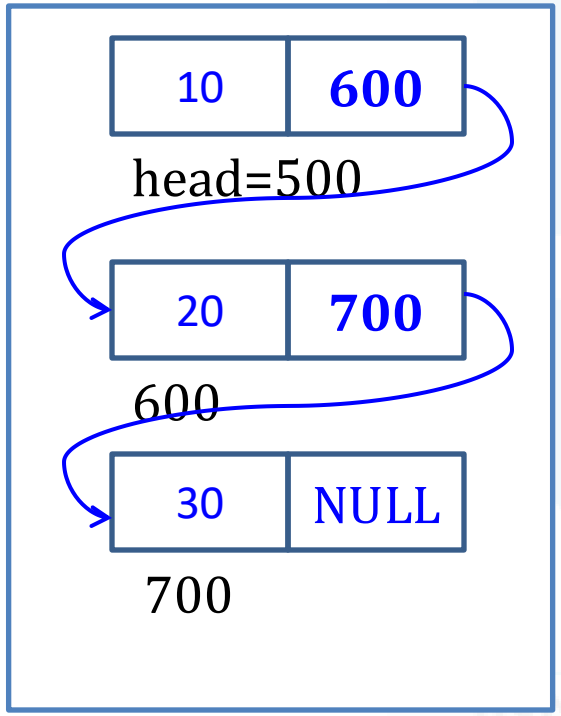


search data =30

```
printf("\nEnter data to be searched: ");
scanf("%d",&data);
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

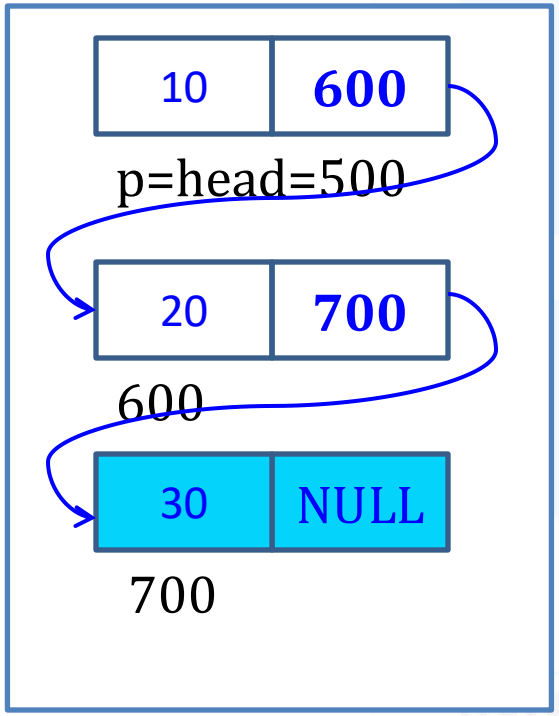
head= search(head);




```
int data, loc=1;
printf("\nEnter data to be searched: ");
scanf("%d",&data);
```

loc=1
search data =30

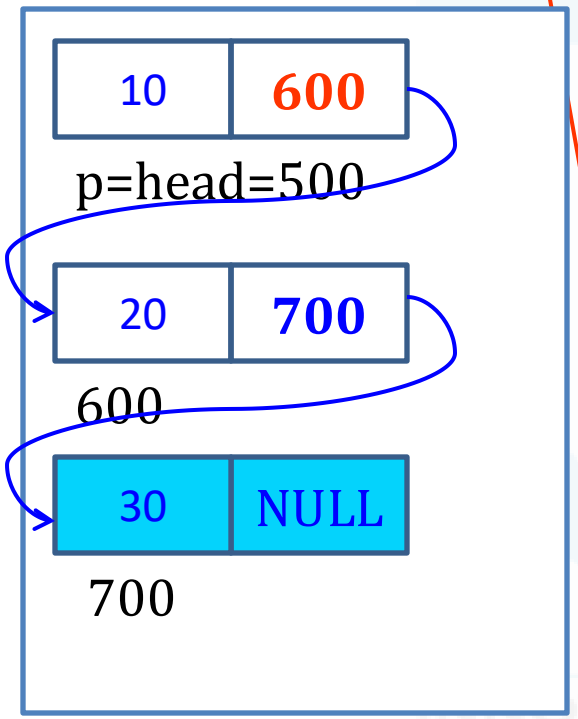
head= search(head);




```
sll *p;
int data, loc=1;
printf("\nEnter data to be searched: ");
scanf("%d",&data);

p=head;
```

loc=1
search data =30

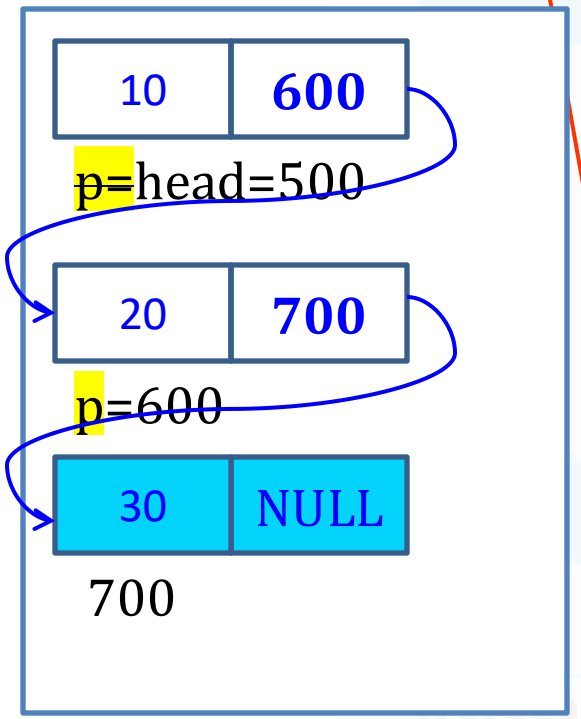
head= search(head);




```
sll *p;
int data, loc=1;
printf("\nEnter data to be searched: ");
scanf("%d",&data);

p=head;
while(p!=NULL && p->data != data)
```

loc=1
search data=30
p->data=10

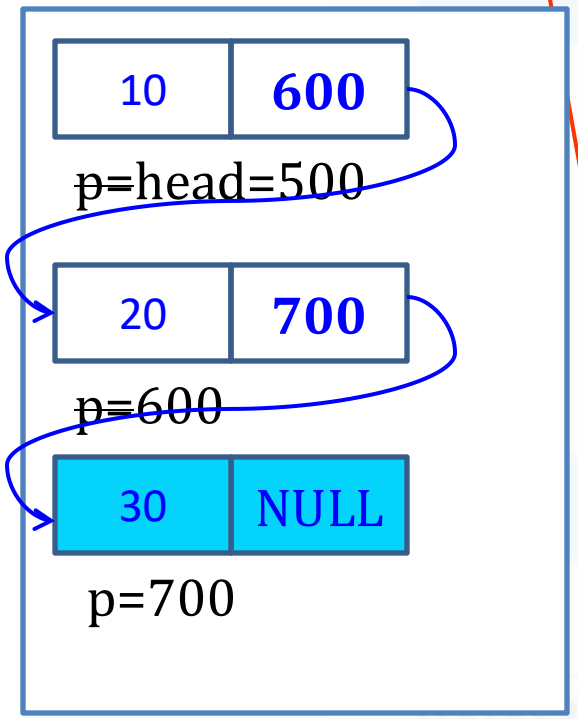
head= search(head);




```
sll *p;
int data, loc=1;
printf("\nEnter data to be searched: ");
scanf("%d",&data);

p=head;
while(p!=NULL && p->data != data)
{ loc++;
p=p->next;
}
```

loc=2
search data=30
p->data=20

head= search(head);



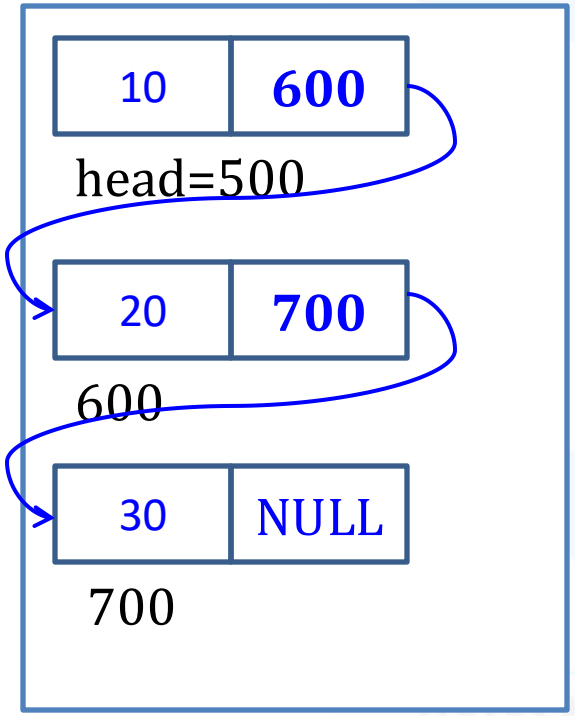
loc=3
search data=30
p->data=30

```
void search(sll *head)
  {       sll *p;
          int data, loc=1;
          printf("\nEnter data to be searched: ");
          scanf("%d",&data);

          p=head;
          while(p!=NULL && p->data != data)
          { loc++;
          p=p->next;
          }

           if(p==NULL)
                printf("\nNot found");
          else
                printf("\nFound at location=%d",loc);
}
```

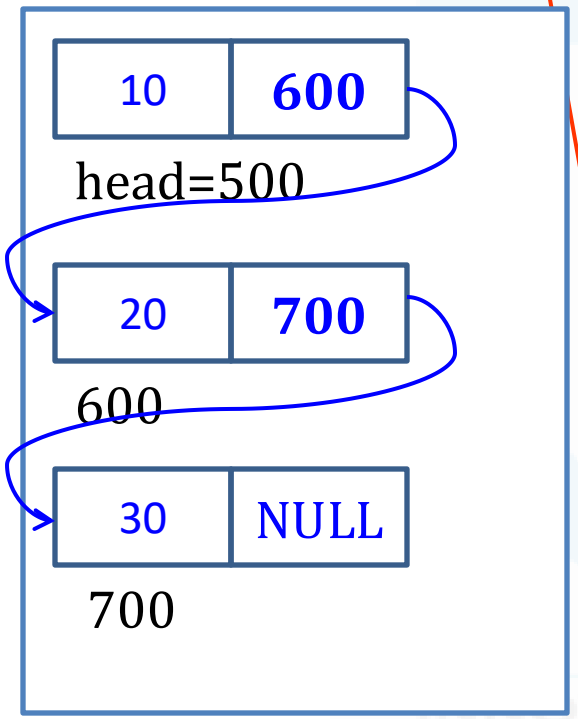
Operations on Singly Linked List (SLL)

- Displaying all nodes from Singly Linked List

head= disp(head);

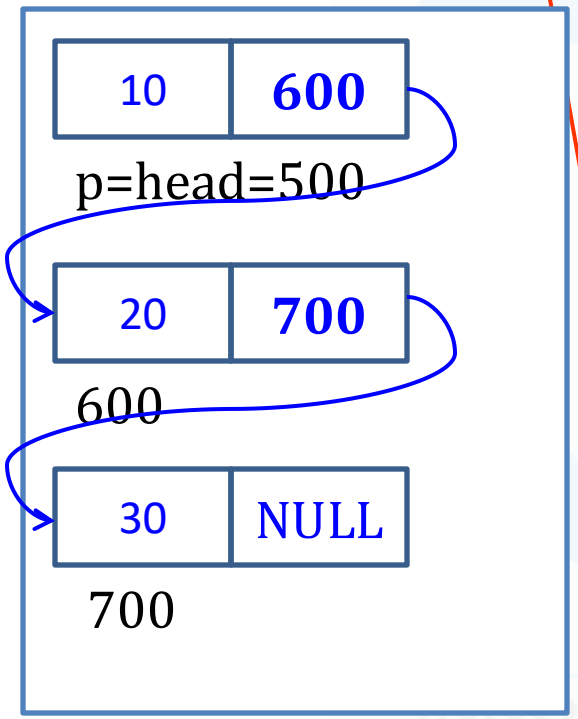


head= disp(head);




```
if(head==NULL)
        printf("\nEmpty Linked List");
```

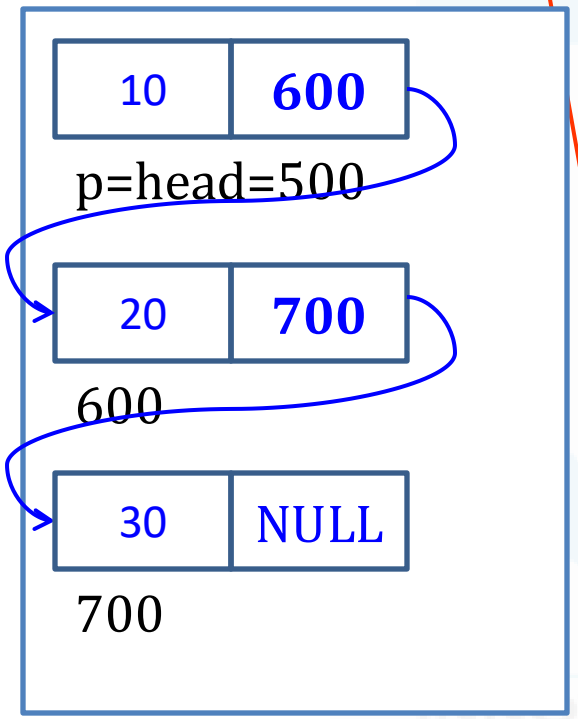
head= disp(head);




```
sll *p;
if(head==NULL)
      printf("\nEmpty Linked List");
else
{
 printf("\n\n Created SLL:\n\n ");
 for(p=head;p!=NULL;p=p->next)
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

head= disp(head);

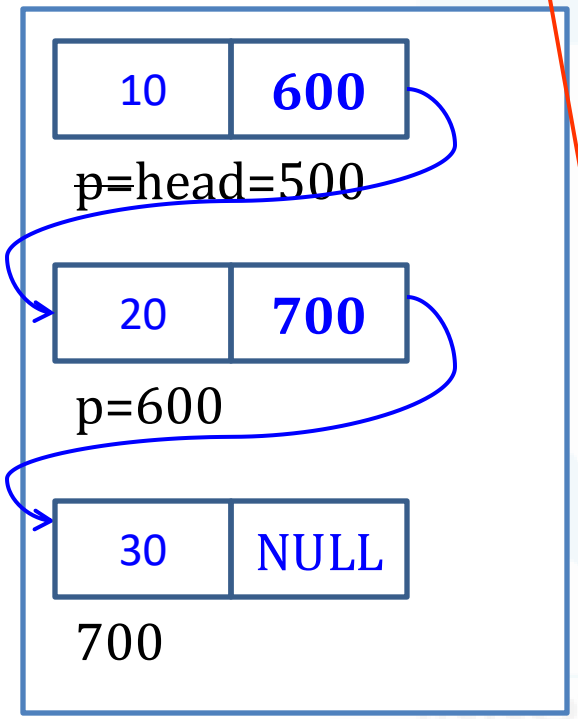



```
void disp(sll *head)
sll *p;
if(head==NULL)
      printf("\nEmpty Linked List");
else
{
 printf("\n\n Created SLL:\n\n ");
 for(p=head;p!=NULL;p=p->next)
      printf("%d->",p->data);
```

10->

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi, Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

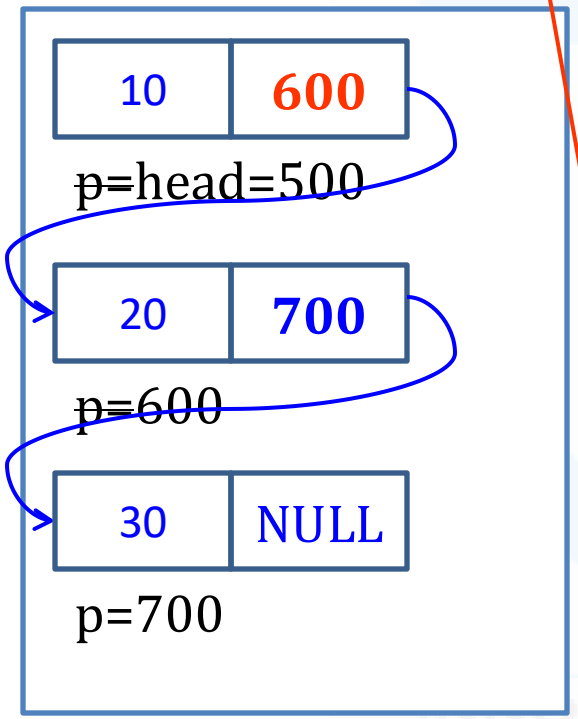
head= disp(head);



10->20->

```
void disp(sll *head)
sll *p;
if(head==NULL)
      printf("\nEmpty Linked List");
else
{
 printf("\n\n Created SLL:\n\n ");
  for(p=head;p!=NULL;p=p->next)
      printf("%d->",p->data);
```

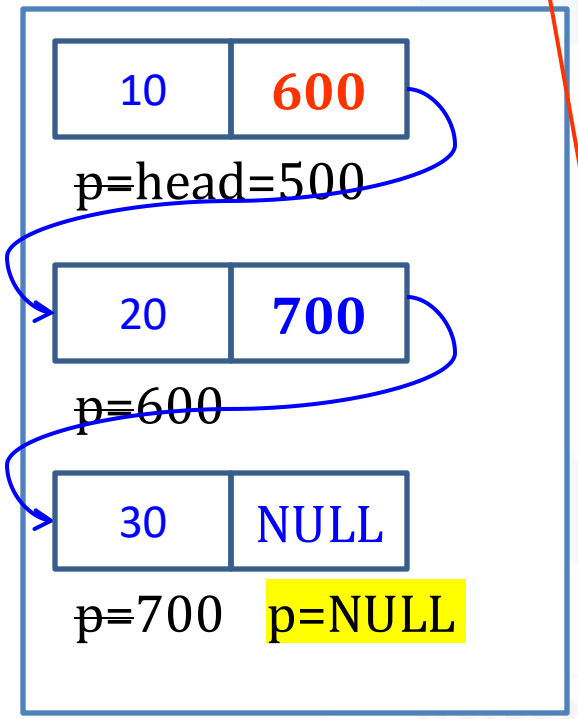
head= disp(head);




```
void disp(sll *head)
sll *p;
if(head==NULL)
      printf("\nEmpty Linked List");
else
{
 printf("\n\n Created SLL:\n\n ");
 for(p=head;p!=NULL;p=p->next)
      printf("%d->",p->data);
```

10->20->30->

head= disp(head);




```
void disp(sll *head)
sll *p;
if(head==NULL)
      printf("\nEmpty Linked List");
else
{
 printf("\n\n Created SLL:\n\n ");
 for(p=head;p!=NULL;p=p->next)
      printf("%d->",p->data);
 printf("NULL");
}
```

10->20->30-> NULL

Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

A Video link

Singly linked list:

- https://log2base2.com/courses/trial-videos/linked-list-basics-trial

Content coverage and depth

4.3 Stack using linked list:

1. Representation of Stack using SLL
2. Functions for Stack operations: push, pop, print

https://www.cs.usfca.edu/~galles/visualization/StackLL.html

4.4 Queue using linked list:

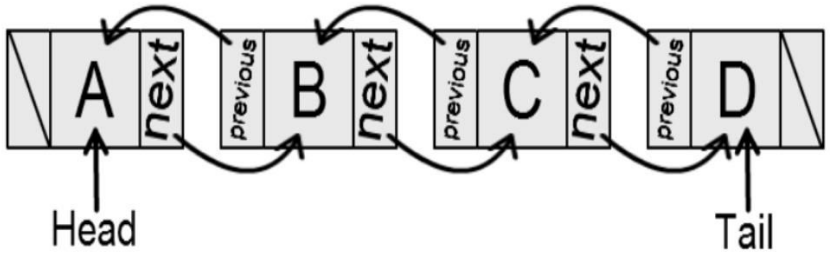
1. Representation of Queue using SLL
2. Functions for Queue operations: insert, delete, print

https://www.cs.usfca.edu/~galles/visualization/QueueLL.html

Content coverage and depth

4.5 Doubly Linked List (DLL):

Representation of DLL



Definition of Node in DLL:

```
typedef struct dll
  { int data;
    struct sll *next, *prev;
  }dll;
```


Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

Content coverage and depth

4.5 Doubly Linked List (DLL):

Advantages of DLL:

- Can be traversed in either direction (may be essential for some programs)
- Some operations, such as deletion and inserting before a node, become easier

Disadvantages of DLL:

- Requires more space
- List manipulations are slower (because more links must be changed)
- Greater chance of having bugs (because more links must be manipulated)

Content coverage and depth

4.5 Doubly Linked List (DLL):

Operations: Creation, Traversal

Content coverage and depth

4.6 Circular Linked List:

1. Representation of Circular Linked List using SLL
2. Representation of Circular Linked List using DLL

Content coverage and depth

Some more animations:

- https://visualgo.net/en/list
- https://yongdanielliang.github.io/animation/web/LinkedList.html

Content coverage and depth

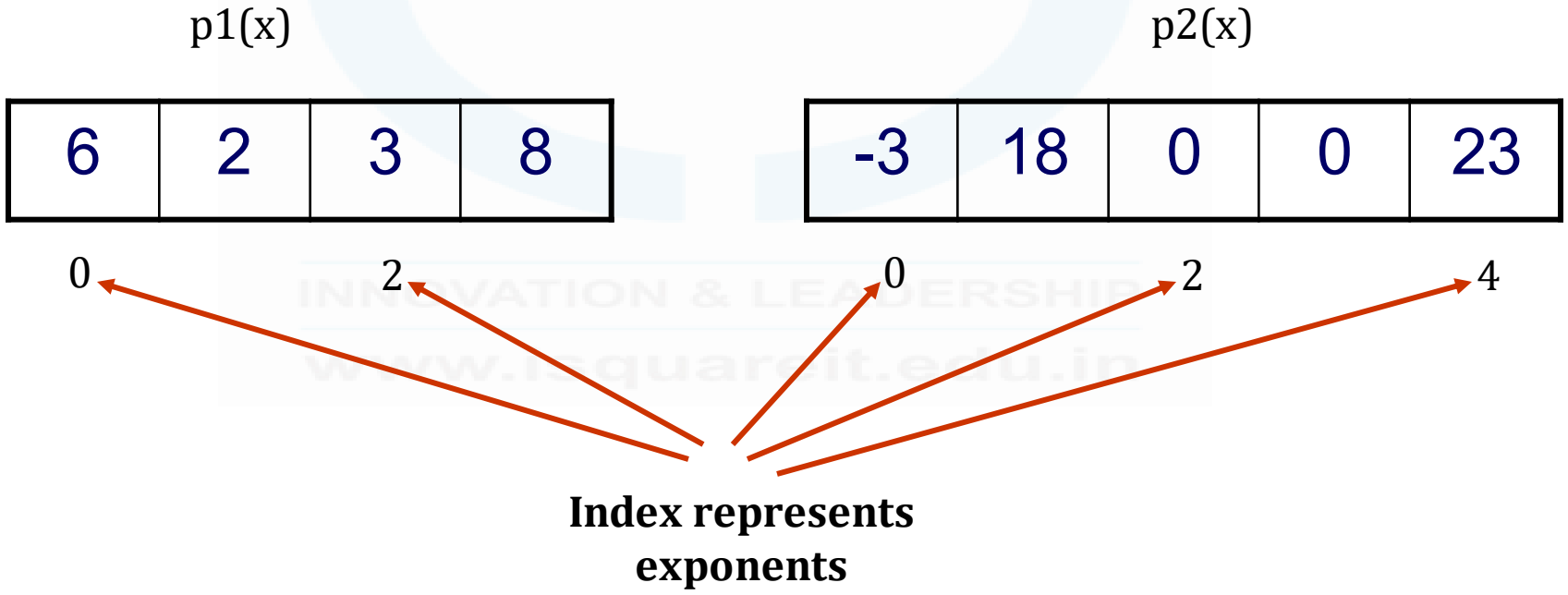
4.7 Representation & manipulations of polynomials using linked list:

1. Advantage of linked list representation for polynomials
2. Node structure for polynomial representation using linked list
3. Manipulation – Addition of polynomials using linked list

Polynomial Representation

Array Implementation:

- $p1(x) = 8x^3 + 3x^2 + 2x + 6$
- $p2(x) = 23x^4 + 18x - 3$



Hope Foundation's International Institute of Information Technology, I²IT, Hinjawadi , Pune - 411 057
Phone - +91 20 22933441/2/3 | Website - www.isquareit.edu.in | Email - info@isquareit.edu.in

Polynomial Representation

- This is why arrays aren't good to represent polynomials:
- $p3(x) = 16x^{21} - 3x^5 + 2x + 6$

6	2	0	0	-3	0	.....	0	16

WASTE OF SPACE!

Polynomial Representation

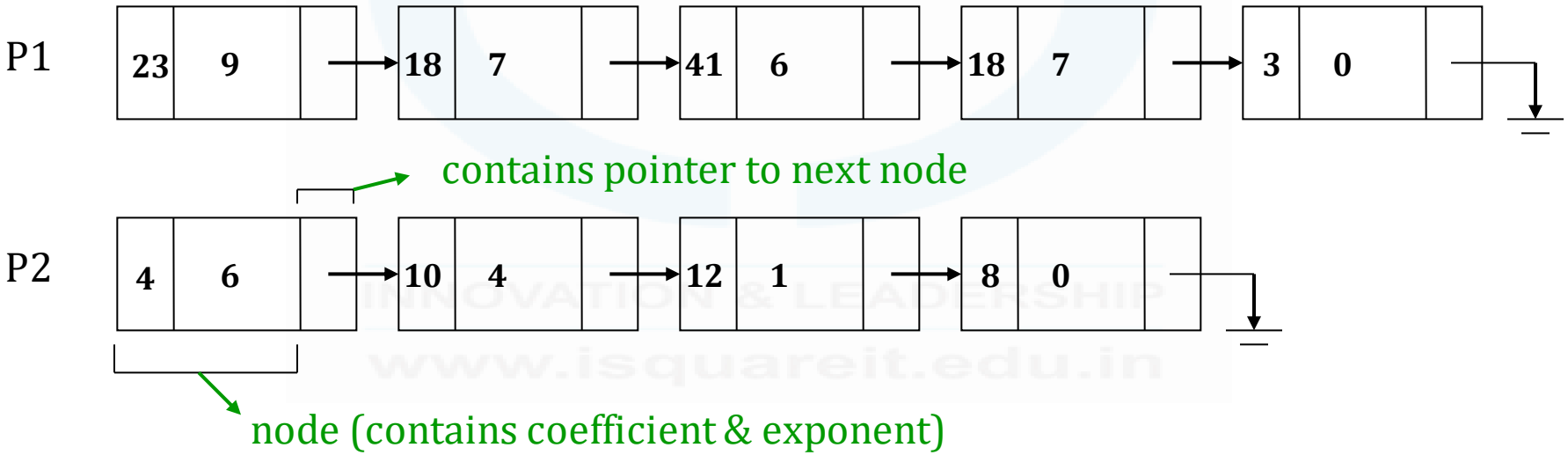
- Advantages of using an Array:
  - only good for non-sparse polynomials.
  - ease of storage and retrieval.

- Disadvantages of using an Array:
  - have to allocate array size ahead of time.
  - huge array size required for sparse polynomials. Waste of space and runtime.

Polynomial Representation

- Linked list Implementation:

$p1(x) = 23x^9 + 18x^7 + 41x^6 + 163x^4 + 3$
$p2(x) = 4x^6 + 10x^4 + 12x + 8$



Polynomial Representation

- Advantages of using a Linked list:
  - save space (don't have to worry about sparse polynomials) and easy to maintain
  - don't need to allocate list size and can declare nodes (terms) only as needed

- Disadvantages of using a Linked list :
  - can't go backwards through the list as SLL is used
  - can't jump to the beginning of the list from the end.

Polynomial Addition

- Adding polynomials using a Linked list representation: (storing the result in p3)

To do this, we have to break the process down to cases:

- Case 1: exponent of p1 > exponent of p2
  - Copy node of p1 to end of p3.

[go to next node]

- Case 2: exponent of p1 < exponent of p2
  - Copy node of p2 to end of p3.

[go to next node]

- Case 3: exponent of p1 = exponent of p2
  - Create a new node in p3 with the same exponent and with the sum of the coefficients of p1 and p2.

Question-1

In the program, we have declared three linked list variables.

i.e struct node *a, *b, *c.

You have to do the following tasks.

Task 1: You need to allocate dynamic memory for those nodes and assign values 100, 200, 300, respectively.

Task 2: Create the following linked list by connecting the nodes.

c->a->b->NULL;

Question-1

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   struct node
   {
          int data;
          struct node *next;
   };
   //declaring nodes
   struct node *a,*b,*c;
   //Implement task 1 here
```

```
   //Implement task 2 here



   //Don't change the below code
   struct node *temp = c;
   while(temp != NULL)
   {
          printf("%d ",temp->data);
          temp = temp->next;
   }
return 0;
}
```

Answer-1

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   struct node
   {
            int data;
            struct node *next;
   };
   //declaring nodes
   struct node *a,*b,*c;
   //Implement task 1 here
   a=(struct node*)malloc(sizeof(struct  node));
   b=(struct node*)malloc(sizeof(struct  node));
   c=(struct node*)malloc(sizeof(struct  node));
   a->data=100;
   b->data=200;
   c->data=300;
   //Implement task 2 here
   c->next=a;
   a->next=b;
   b->next=NULL;
   //Don't change the below code
   struct node *temp = c;
   while(temp != NULL)
   {
            printf("%d ",temp->data);
            temp = temp->next;
   }
return 0;
}
```

More Questions

1. There is a singly linked list with n nodes, each node contains integer data, your task is to identify count of each number present into the list with suitable Counting algorithm.

2. A singly linked list contains n numbers, few of them repeats, your task is to create one more list which contains only UNIQUE elements by writing suitable algorithm

More Questions

3. A singly linked list contains n numbers, few of them are odd and others even, your task is to write an algorithm which will separate them into odd and even and write it to two different lists
4. In the doubly linked list, each node points to both its predecessor and its successor. There is a naughty boy who mis-pointed the head node to some other node(different from the second node). Your task is to identify mis-pointed node with an algorithm

Platform open for discussion

THANK YOU!