DATE : __/__/__
PAGE : _____

Lecture 9 Linked List

- Linear Data structure
- Linear order maintained using pointer (link)

* Advantages of Array

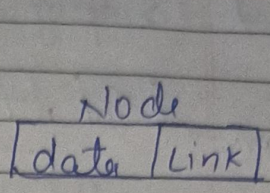
- elements can be accessed randomly by using the index
- using Array, other data structure like linked list, stacks, queues, tree graphs etc can be implemented
- Two 2d array are used to represent metrices

* Disadvantages of Array

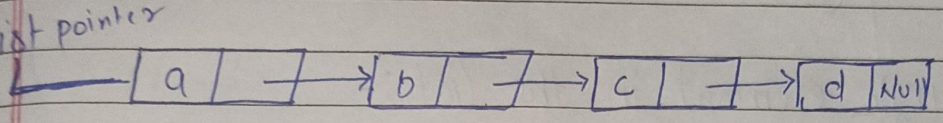
- The number of elements to be stored in an array should be known in advance
- Allocating more ~~no~~ memory than the requirement leads to wastage of memory space and less allocation of memory also leads to a problem
- An array is a static structure. Once declared the size of the array cannot be modified
- Insertion and deletion are quite difficult

DATE : __/__/__
PAGE :

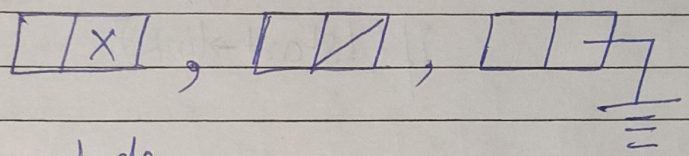
- List contains node
  Node contain 2 fields
  - Data (key or value)
  - Link (Next or down)

Node


- There is list pointer which points to the first node of list



- Representation of NULL



* Accessing Node

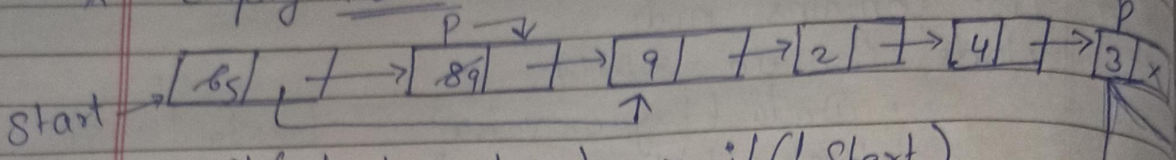
list → data
list → link

printf("%c", list → data); = a
printf("%c", list → link → data); = b

* Node structure

```
Struct node {
    char data;
    Struct node *link;
};
```

* Empty list Condition



if (start == NULL) or if (!start)

- not empty list

if (start != NULL) or if (start)

- Single element in the list :-

if (start -> link == NULL)

or

if (!start -> link)

* Null pointer Dereferencing

Same list

printf("%d", p -> link -> data); => Null pointer dereferencing

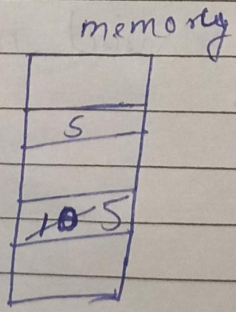
NULL

* Link ~~sto~~ Assignment

int a = 5, b = 10;

b = a;

Location value



DATE : __/__/__
PAGE : _____

Start → data = 5

Same list

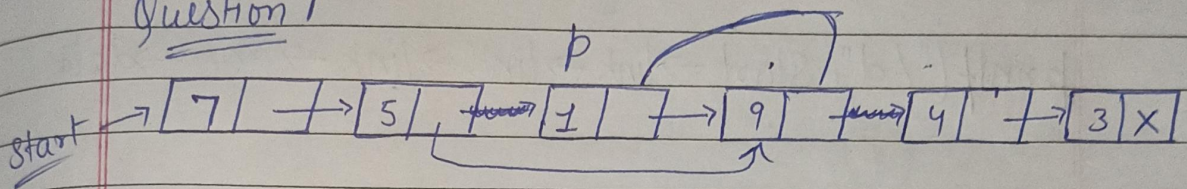
p = Start → link

p → data = p → link → data
(9)

p → 1.

Start → link = p → link

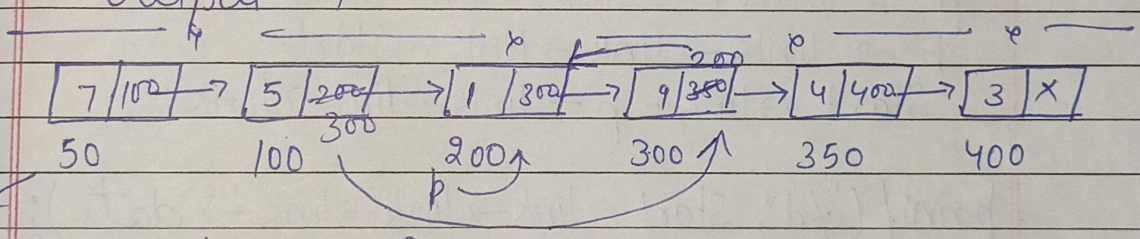
Question 1



struct node *p;
p = start → link → link;
(start → link) → link = p → link;
(p → link) → link = p;

printf("%d", (start → link → link → link → link) → data);

output = 9

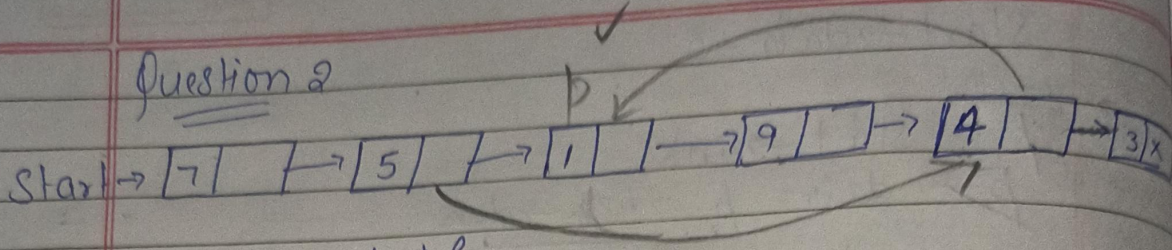


struct node *p;
p = start → link → link; — p = 200
start → link → link = p → link; → 300
p → link → link = p; → 200

printf("%d", start → link → link → link → link → data);

output = 9

Question 2

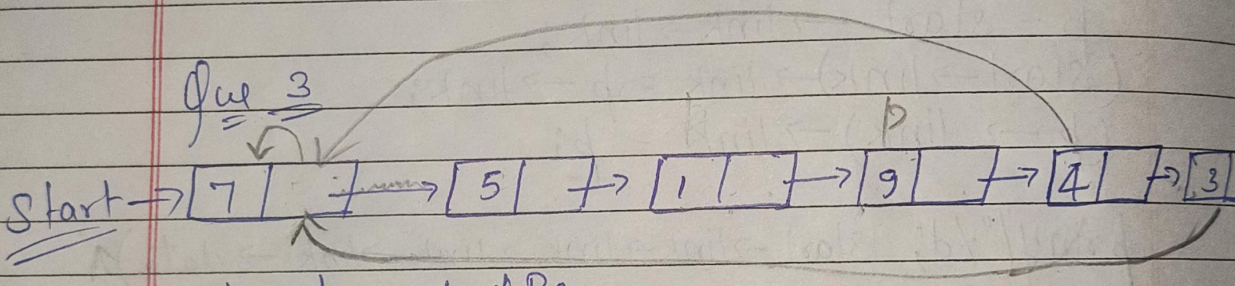



```
struct node *p;
p = start -> link -> link;
(start -> link) -> link = p -> link -> link
(p -> link -> link) -> link = p
```

```
printf("%d", start -> link -> link -> link -> link -> data
```

(1)

Que 3



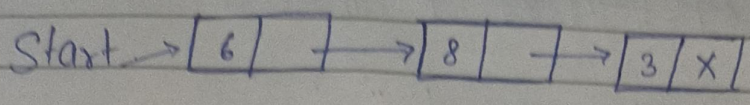

```
struct node *P;
p = start -> link -> link -> link
p -> link -> link -> link = start
start -> link = p -> link -> link -> link
```

```
printf("%d", start -> link -> link -> link -> data );
```

~~5~~ (7)

DATE : __/__/__
PAGE :

Lecture 10 - Linked list - Traversing and Insetion




```
struct node *p = start
while (p != NULL) {
    process  p -> data
    p = p -> link
}
```

Run time complexity = O(n)

* Finding the adress of last node

```
p = start
if (p == NULL) return;
while (p -> link != NULL)  or  while (p -> link)
{
    p = p -> link
}
return p;
```

- Run time complexity = O(N)

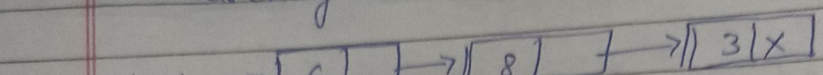
* Counting nodes in list

```
int count = 0;
p = start
while (p != NULL) {
    count++;
    p = p -> link;
}
return count;
```

Time Complexity = O(N

DATE : __/__/__
PAGE : ____

- Searching in Linked list



- only linear search because linked list is not stored sequence wise so we can't get the mid element to perform "Binary Search"

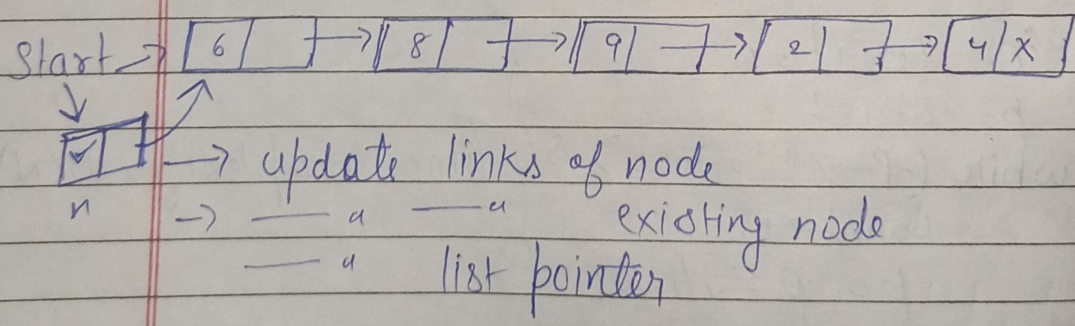
```
p = start
while (P!=NULL) {
    if (p -> data == item)
        return p
    p = p -> Link
}
return NULL;
```

Time Complexity = O(n)

* Insertion in linked list

- Creating a New Node

```
struct node *n = (struct node*) malloc (sizeof(struct node))
```



→ update links of node
→ " " existing node
" " list pointer

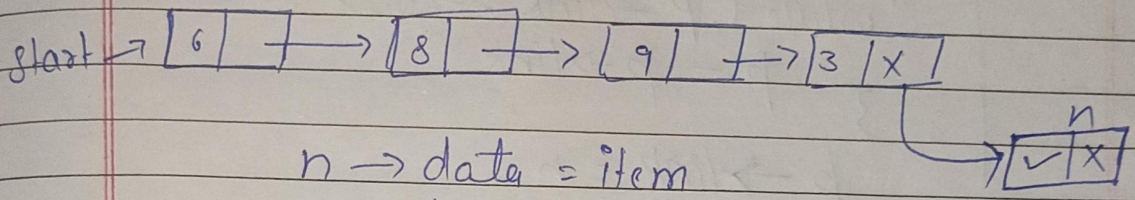
DATE : __/__/__
PAGE :

insertion_at_bgnning(start , item) {

```
n → data = item
n → link = start
start = n
}
```

Time Complexity = O(1)

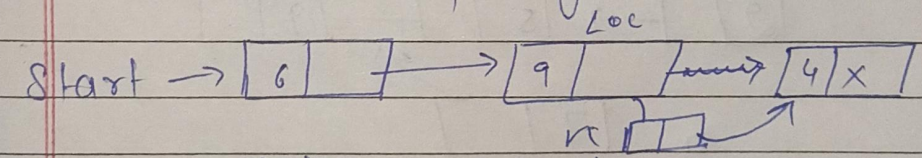
* Insertion at the end




```
n → data = item
n → link = NULL
if (start = = NULL) start = n;
                    return;
p = start
while (p != NULL) {
        p = p → link;
}
p → link = n
```

Time complexity = O(

* Insertion After given node

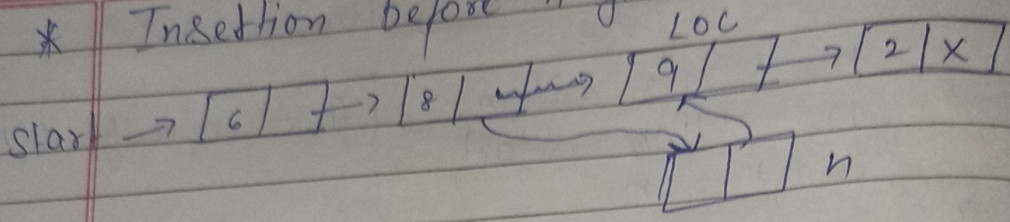


Time complexity ~~O(n)~~ O(1)

```
n → data = item
n → link = loc → link
loc → link = n
```

DATE : / /
PAGE :

* Insertion before A given Node

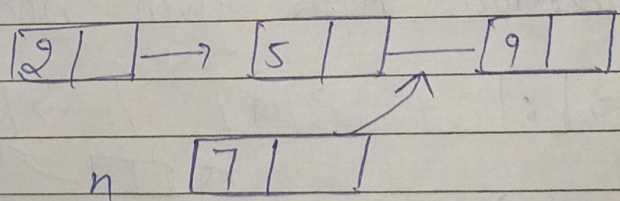


Time Complexity
O(n)

```
n -> data = item
n -> link = loc
if (loc == start) start = n; return
p = start
while (p -> link != loc) {
    p = p -> link
}
p -> link = n
```

Quiz① What is the complexity to insert a new node in a stored linked list

A O(1)
X b O(log n)
c O(n) ✓
X d O(n log n)



We need traverse the linked list and compare the n with existing node and according the we need to insert the element, Hence it will take O(N) time

DATE : __/__/__
PAGE : _____

(2) what is the complexity to find k^{th} node from starting in a singly linked list?

A O(n) ✓