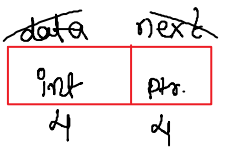
Linked List: is collection of specially designed element call as node.

Wednesday, June 21, 2023 9:15 AM

Node is element of linked list which can have at least 2 members

1. Data
2. Address of another element



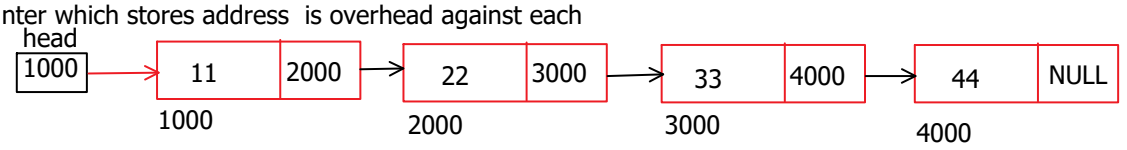
Advantages:

1. Optimized usage of memory
2. Size(no.of elements) of linked list can be increased or decreased at runtime

Int arr[4] = 16

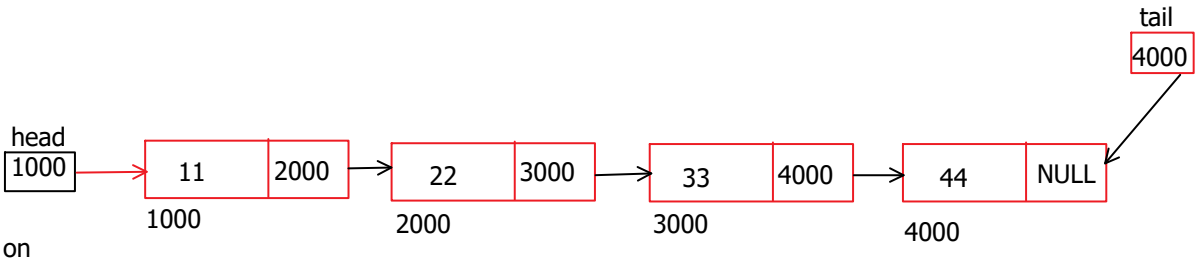
Disadvantages:

1. In case of linkedlist each node maintains address of another element. So memory given for that pointer which stores address is overhead against each node
2. Traversal is combersome



Types of Linked List:

1. Singly Linear Linked List
2. Singly Circular Linked List
3. Doubly Linear Linked List
4. Doubly Circular LinkedList

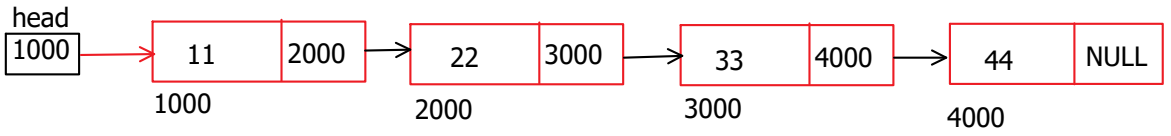


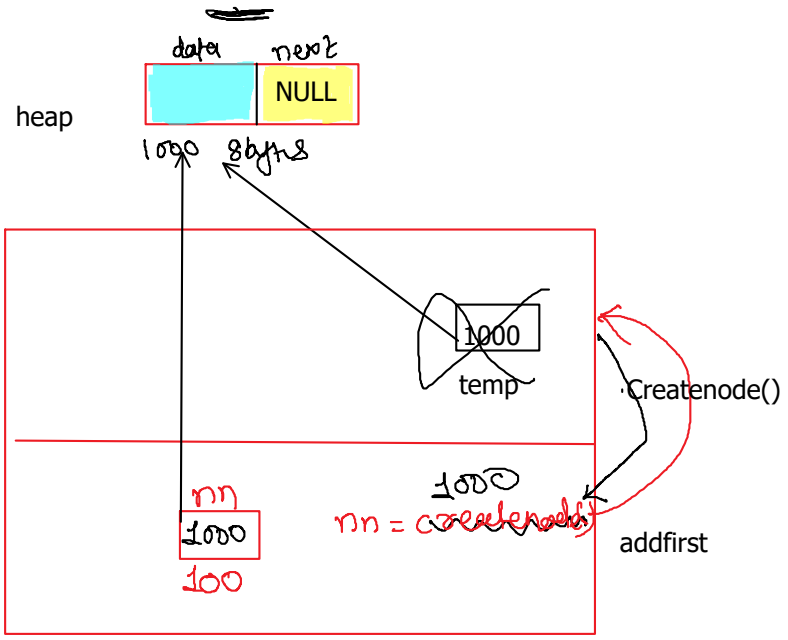
Operations can be performed on Linked List

1. Addatfirst,addatlast,addatpos
2. Delfirst,dellast,delfrompos
3. Traverse
4. Reverse
5. Merge
6. Sort
7. Etc.

```
struct node
{
    int data;
    struct node *next;
};
```

Self referential structure is a structure which has atleast one member as pointer which points self type in which it is declared






```c
node_t * create_node()
{
    node_t *temp;
    temp = (node_t *)malloc(sizeof(node_t));
    temp->next = NULL;
    return temp;
}
```

temp -> data

temp -> next (*temp).next

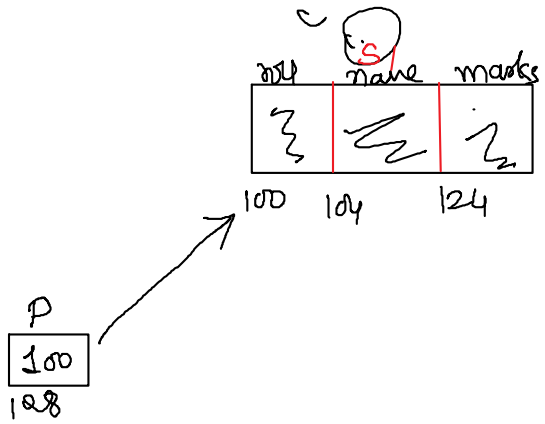
1000

int *

node_t *

address of int

address of node



struct student *p;

p = &s1

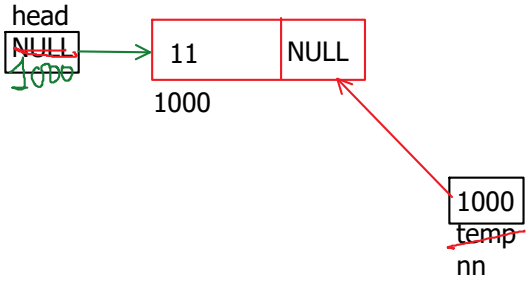
s1.rollno

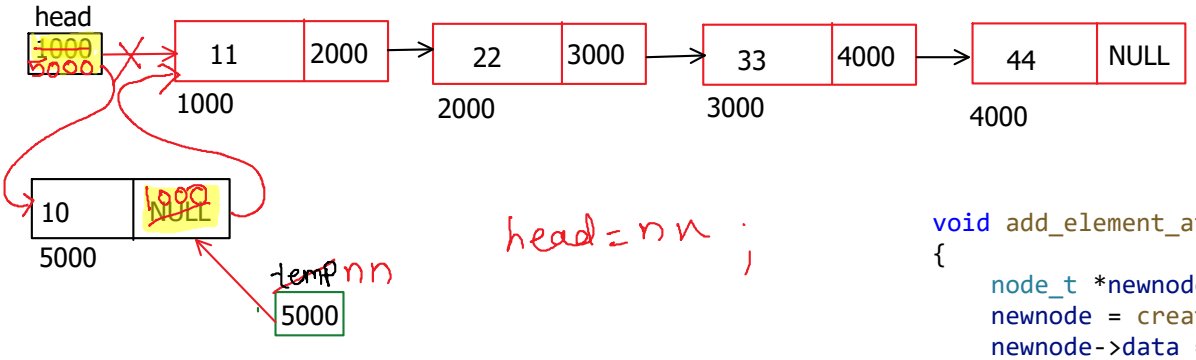
s1.sname

p->rollno

p->sname

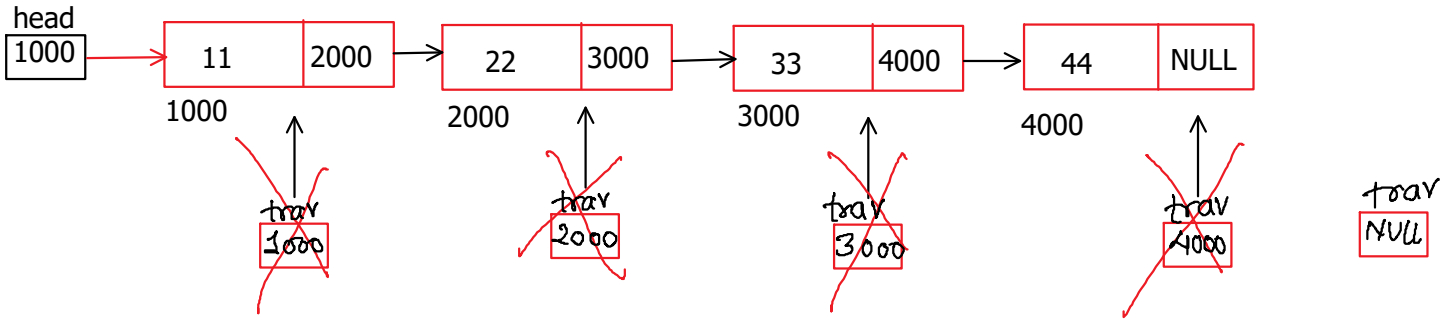
p->marks





head = nn ;

```
void add_element_at_first(int *p)
{
    node_t *newnode;
    newnode = create_node();
    newnode->data = *p;
    if(head ==NULL)
        head = newnode;
    else //3.2 if list is ready
    {
            newnode->next = head;
          head = newnode;
    }//3.2.1. attach node
}
```



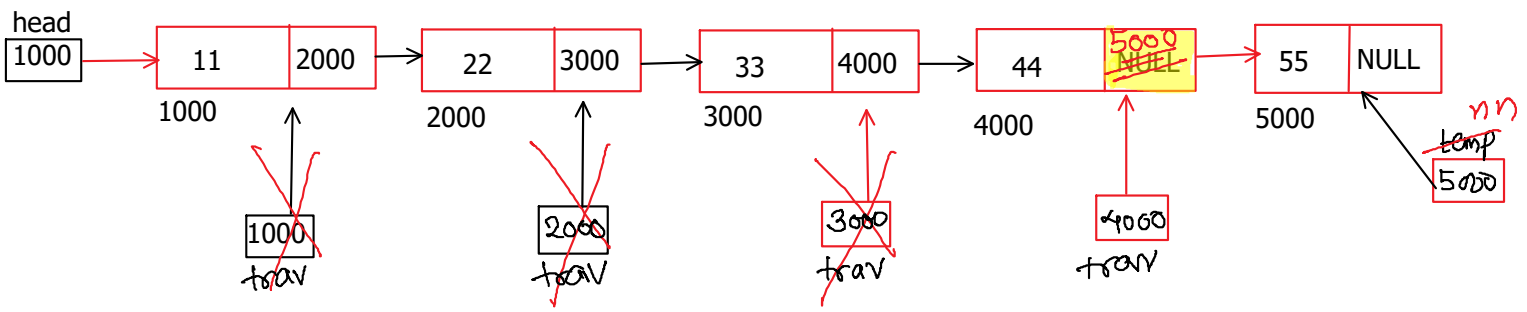
11 -> 22 -> 33 -> NULL

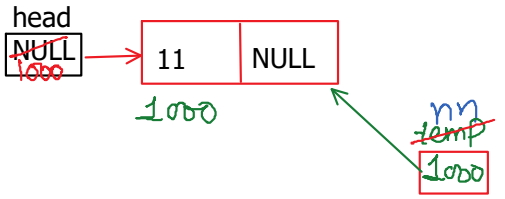
```
node_t  *trav = head;

while(trav!=NULL)
{
    printf("%d -->",trav->data);
    trav=trav->next;
}
```

Wednesday, June 21, 2023 9:58 AM

trav->next = nn

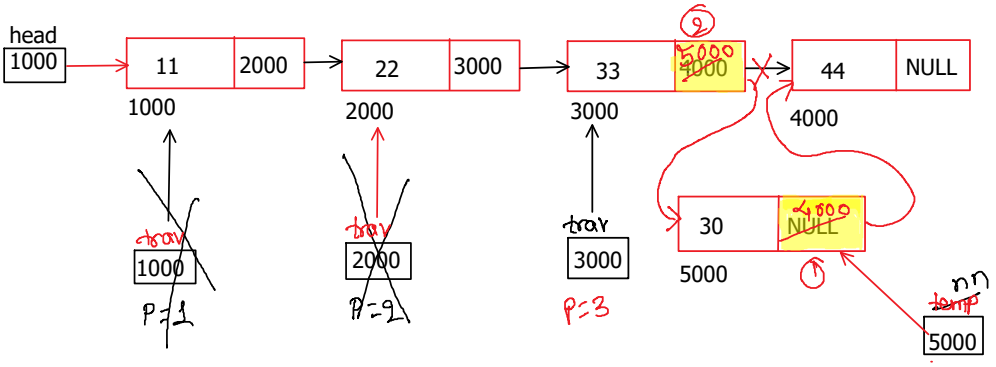




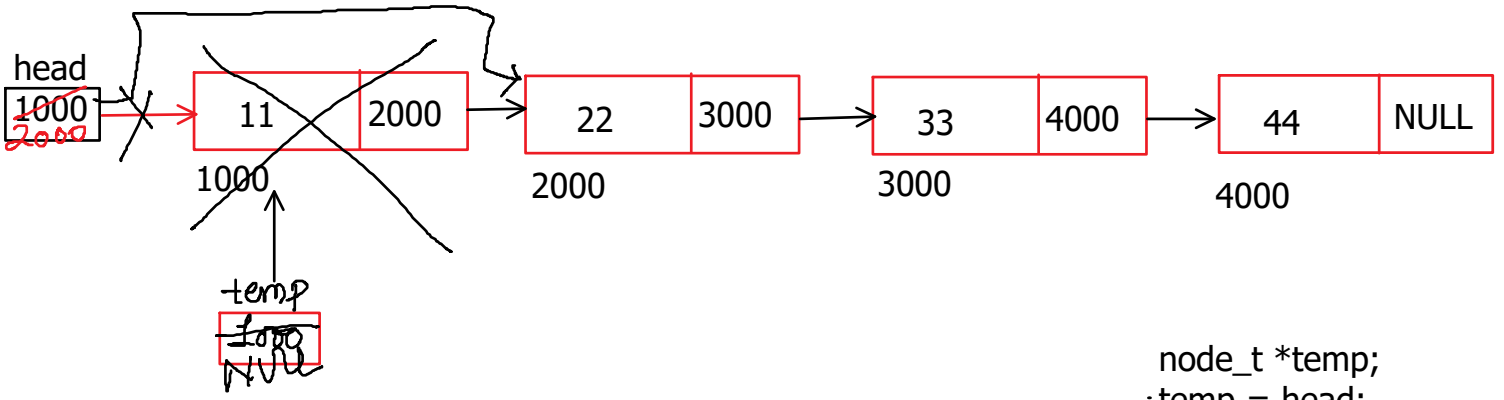

```
void add_element_at_last(int *p)
{
    node_t *newnode,*trav;
    newnode = create_node();
    newnode->data = *p;
    //3. attach node in collection/list
    if(head ==NULL)
        head = newnode;
    else
    {
        trav=head;
        while(trav->next!=NULL)
        {
            trav=trav->next;
        }
        trav->next = newnode;
    }
}
```

Wednesday, June 21, 2023 9:58 AM

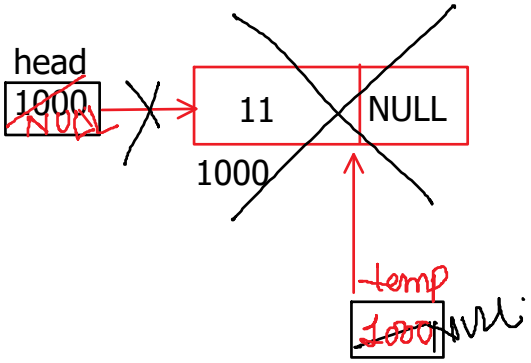
Pos=4

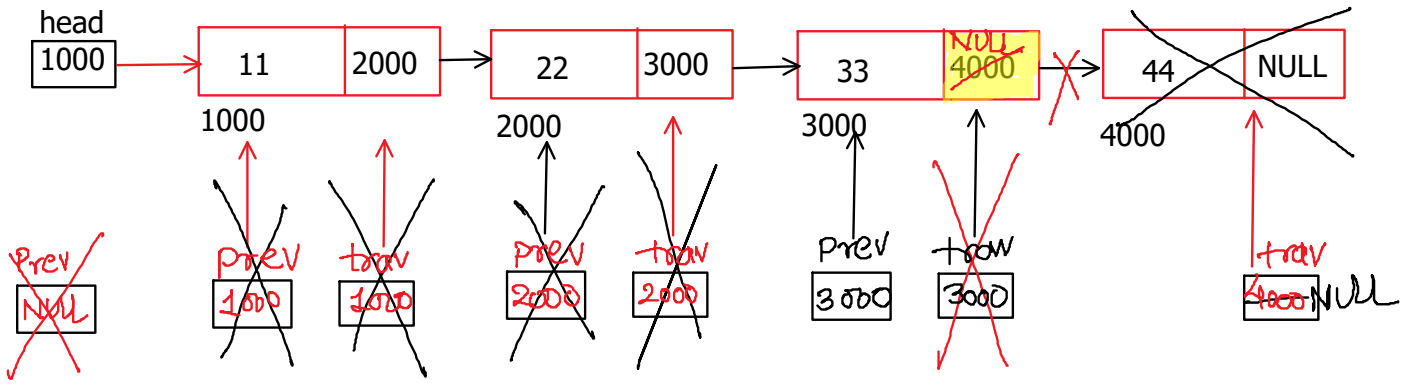


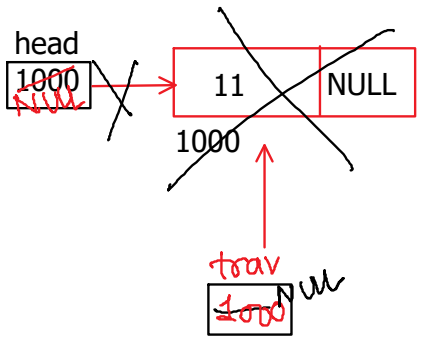

```
void add_element_at_pos(int *p,int pos)
{
    node_t *newnode,*trav;
    int p;
    if(pos == 1)
        add_element_at_first(p);
    else if(pos == size()+1)
        add_element_at_last(p);
    else
    {
        newnode = create_node();
        newnode->data = *p;
        trav=head;
        p = 1;
        while(p < pos-1)
        {    3 < 3
            trav=trav->next;
            p++;
        }
     newnode->next = trav->next;
     trav->next = newnode;
    }
}
```




```
node_t *temp;
temp = head;
head=head->next;
free(temp);
temp=NULL;
```



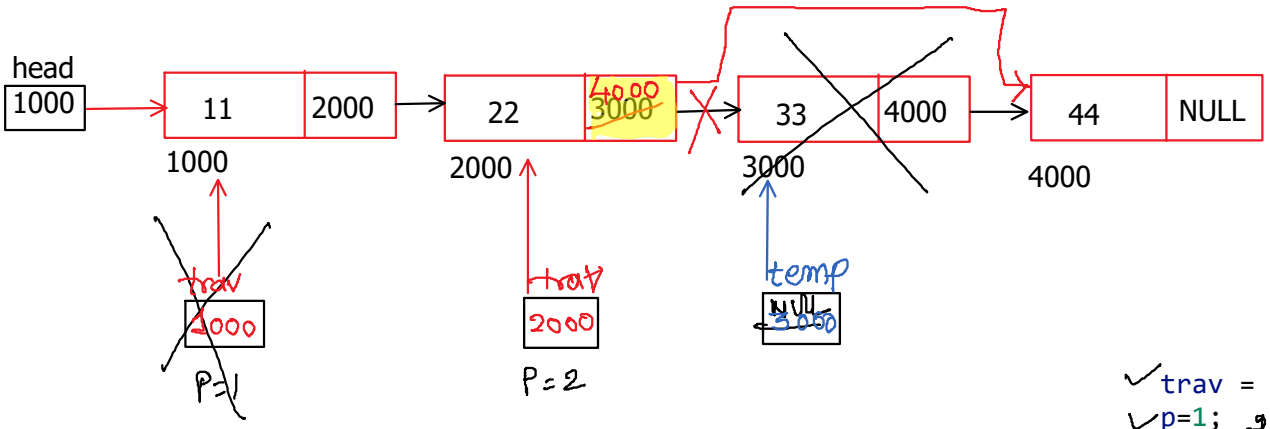




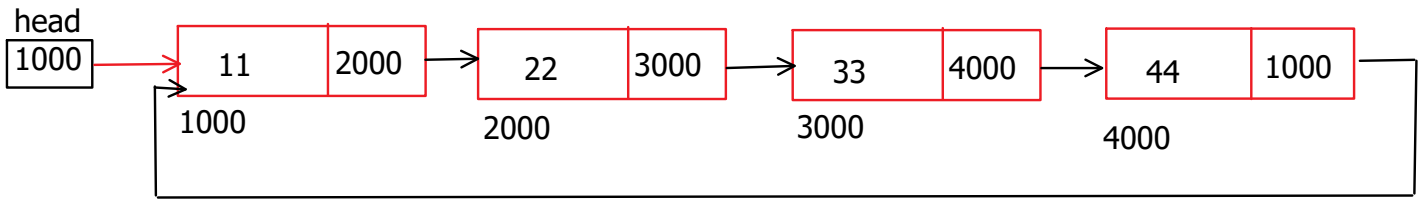

```
trav=head;
If(head->next==NULL)
    Head=NULL;
Else
{
    while(trav->next!=NULL)
    {
        prev = trav;
        trav=trav->next;
    }
    prev->next=NULL;
}
free(trav);
trav=NULL;
```

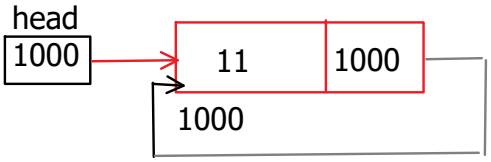
POS = 3

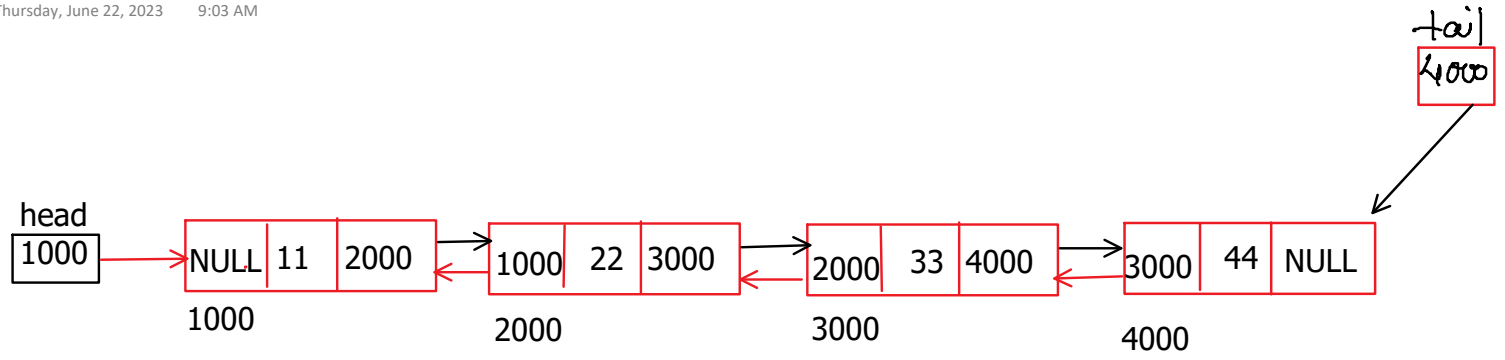
Wednesday, June 21, 2023 9:58 AM




```
trav = head;
p=1;
while(p < pos-1)
{
   trav=trav->next;
   p++;
}
temp = trav->next;
trav->next=temp->next;
free(temp);
temp=NULL;
```


head
1000
11
2000
1000
22
3000
2000
33
4000
3000
44
1000
4000


head
1000
11
1000
1000




```
struct node
{
    struct node *prev;
    int data
    struct node *next;
};
```

Thursday, June 22, 2023 9:19 AM

Thursday, June 22, 2023
9:03 AM