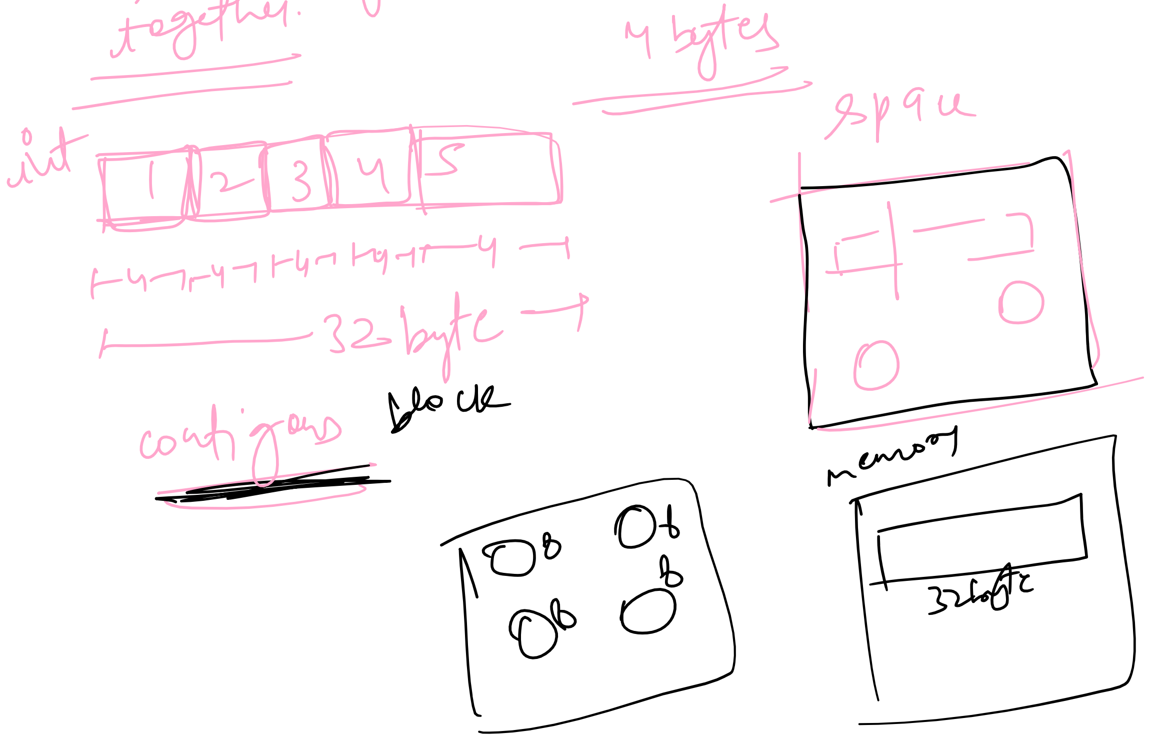
Day 13

23 November 2022 20:52

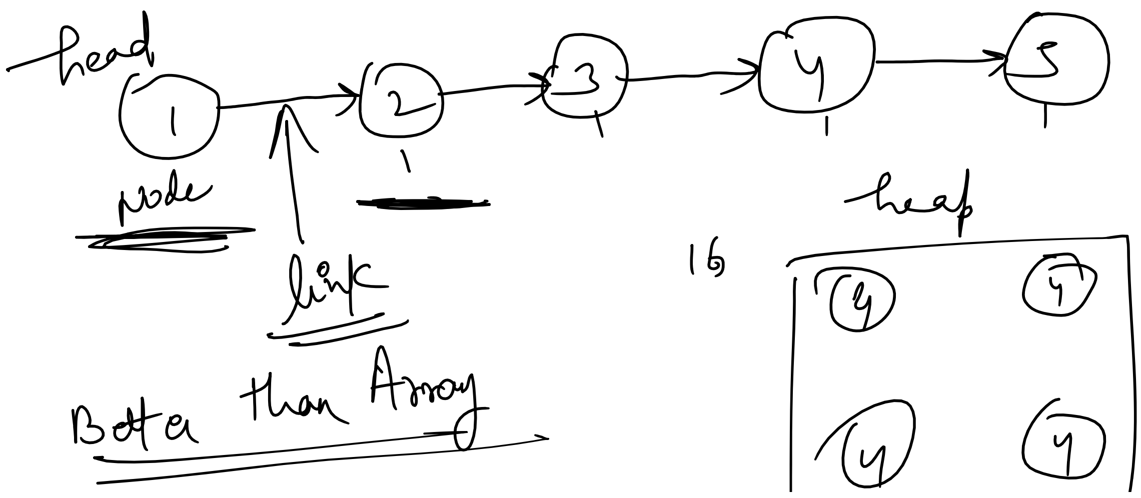
LINKED LIST

① linked list

Collection of Nodes that are linked together.



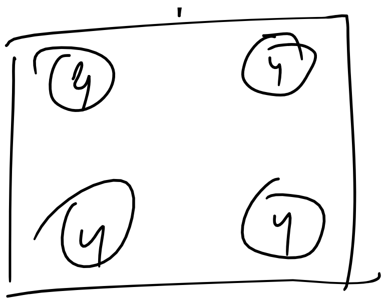
LINKED LIST



Better than Array

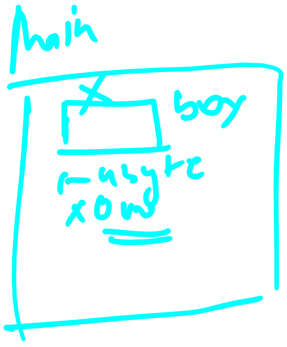
link

Better than Array



POINTERS 4byte

int x;

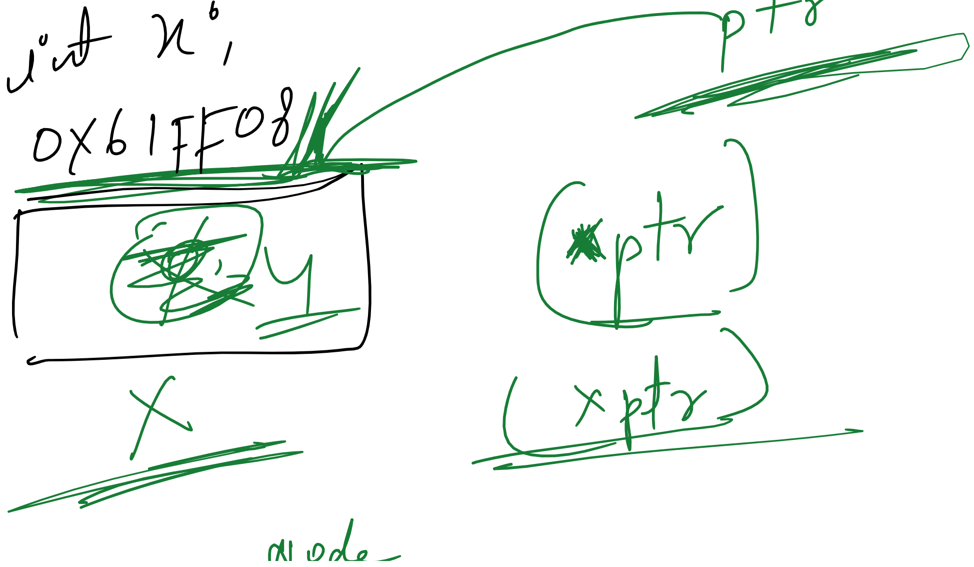


pointer variable का Address store करता है।

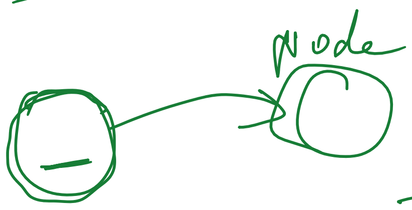
int*   float*

double*

int* ptr = &x;



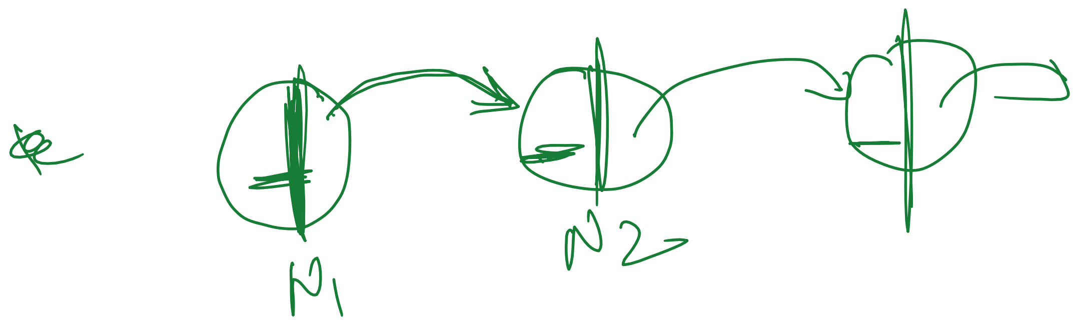
x (xptr)



class

```
struct Node {
    int val;
    Node* next;
}
```

```
class Node {
    public:
    int val;
    Node* next;
};
```

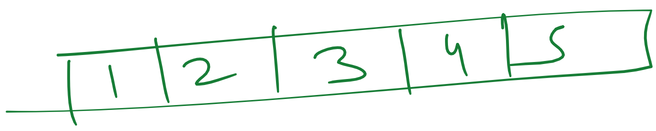


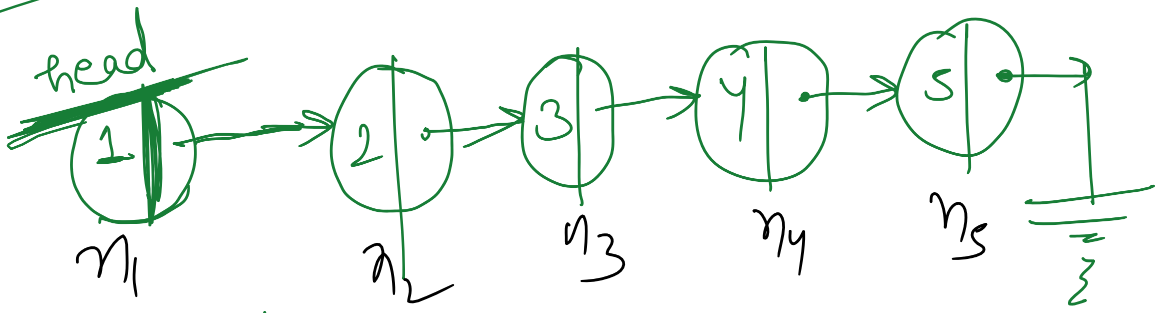
Create

```
class Node {
    public:
    int value;
    Node* next;
}
```

heap

kartik

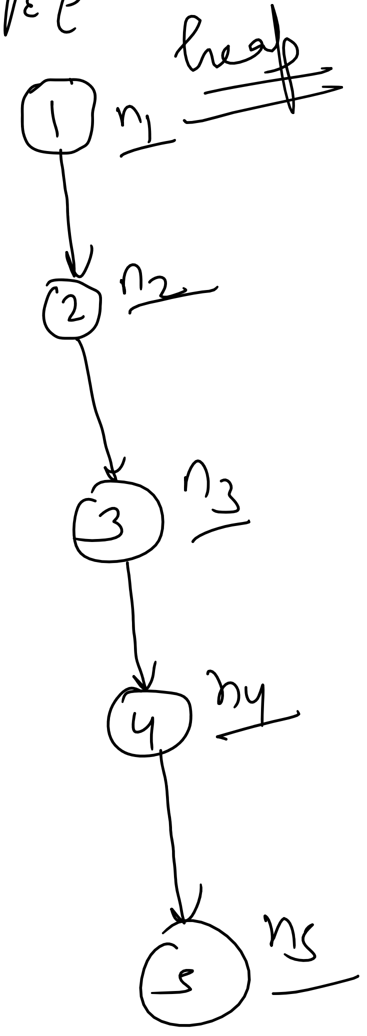




Create Code
C/C++

// Dynamically memory allocate करना आना चाहिए

```
Node* n1 = new Node();
   n1->val = 1;
Node* n2 = new Node();
   n2->val = 2
   n1->next = n2
Node* n3 = new Node();
   n3->val = 3
   n2->next = n3
Node* n4 = new Node();
   n4->val = 4
   n3->next = n4
Node* n5 = new Node();
   n5->val = 5
   n4->next = n5
```

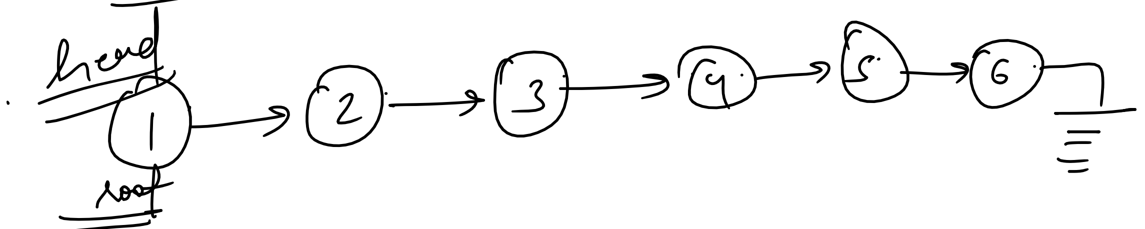


Nodes X

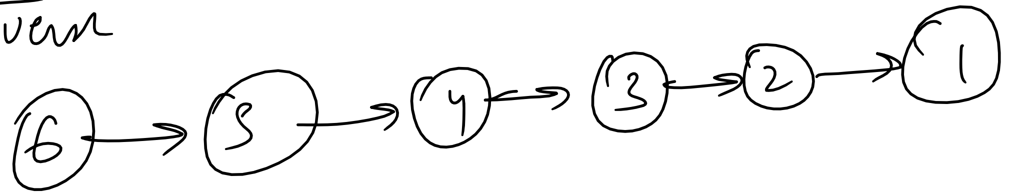
Dynamic Memory Allocation (2)

Pointer (1)

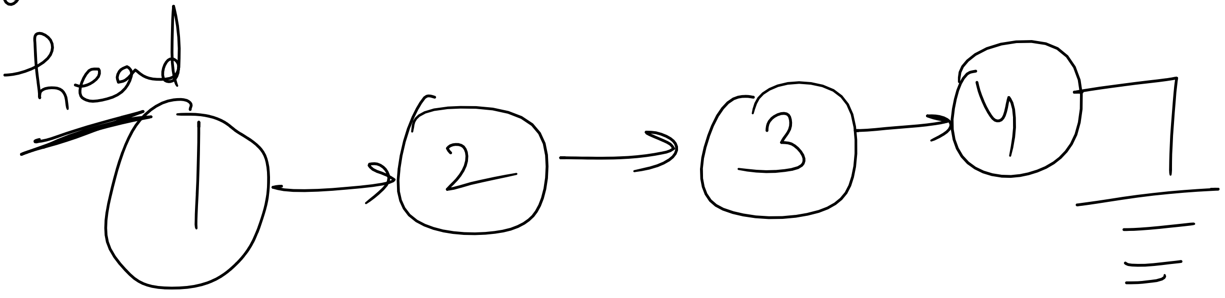
Linked List Creation



Reverse



70/80 linked list } Imagination



Print

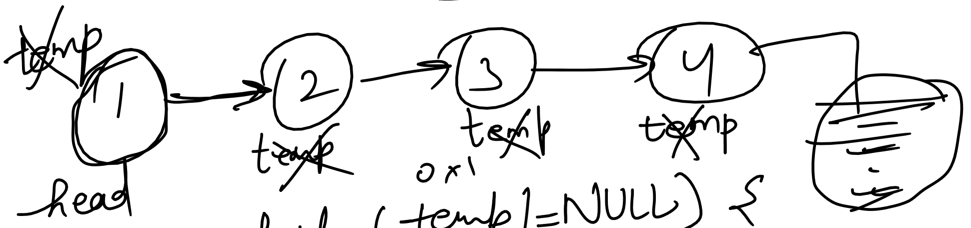
* head को कभी अपनी जगह से remove नहीं करना ||

* Traverse करने के लिए दूसरा Pointer use करना ||

void PrintLinkedList (class Node * head){

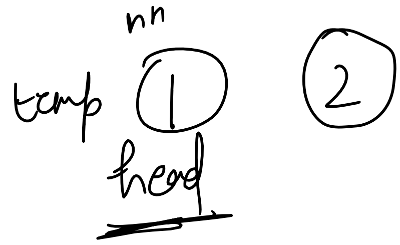
change नहीं करना

class Node* temp = head;

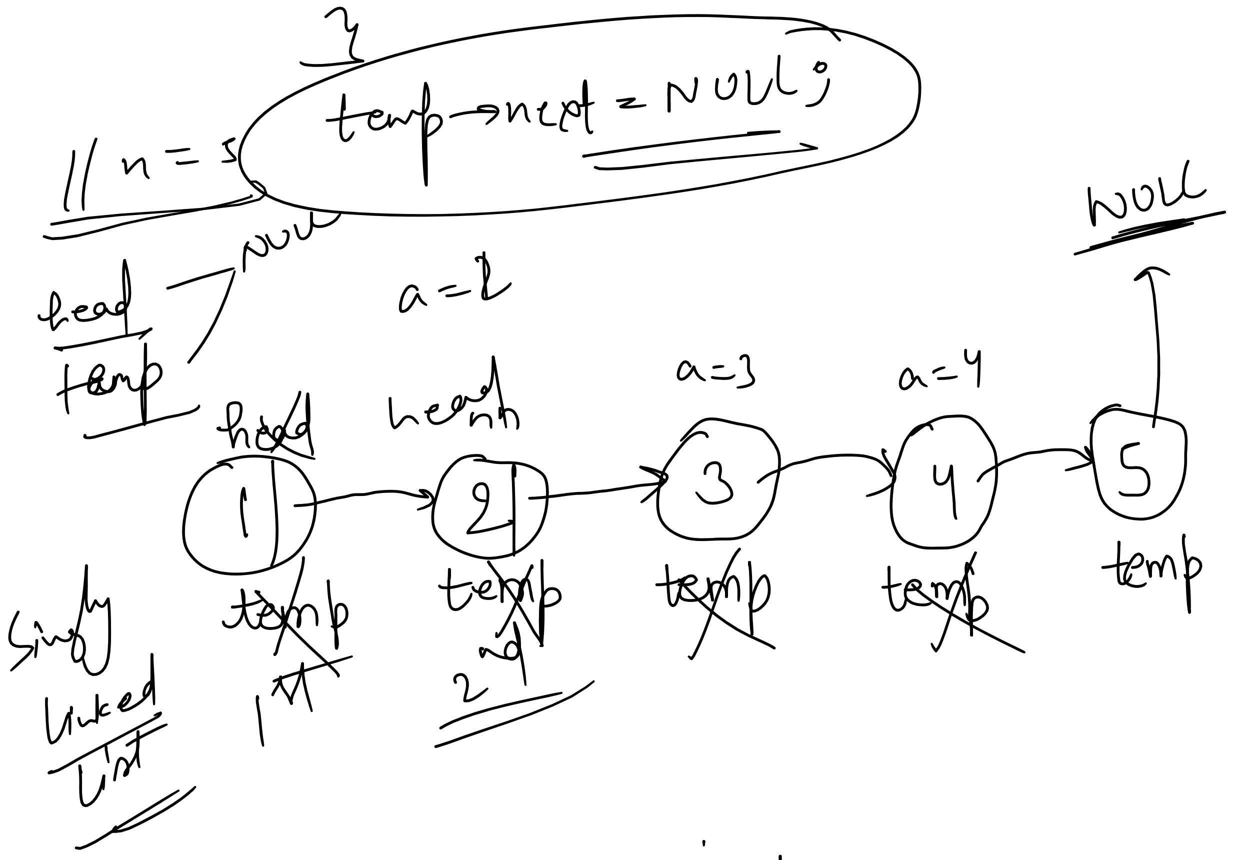


```
while (temp!=NULL) {
    cout << temp->val << endl; // 1 2 3 4
    temp = temp->next;
}
```

Linked list



```
int main() {
   int n;
   cin>>n;   //n=5
 Node *head = NULL; // pointer safety
 Node *temp=NULL
 for(int i=0; i<n; i++){
        - int a;
        - cin>>a;
    Node* new_node = new Node();
      new_node->val = a;
      if (head == NULL)
       { head = new_node; temp=new_node;
       }else{
           temp->next= new_node;
       } temp= new_node;
```



① 2nd position पर insert

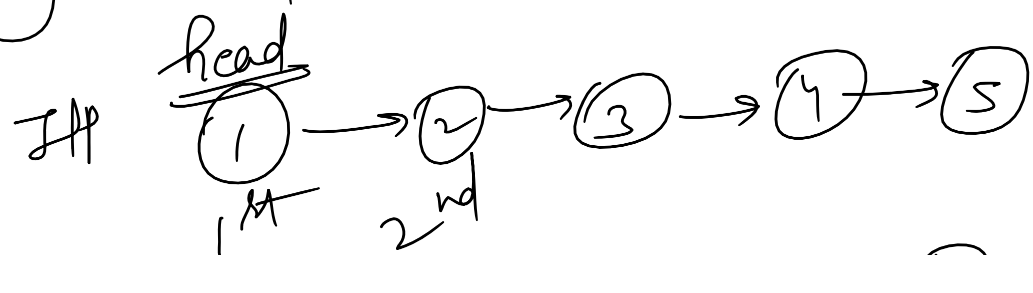
② Delete 4th position वाला Element

③ 1st Node को remove

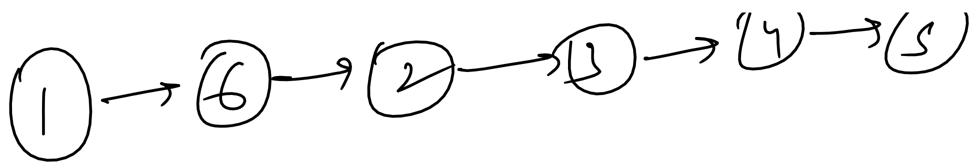
④ last Node node remove

⑤ 1st position पर insert

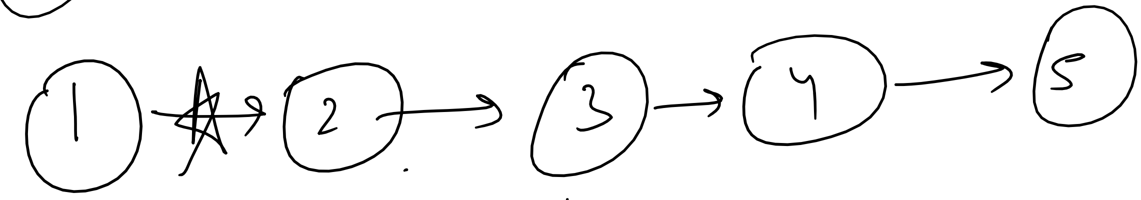
⑥ last position पर insert.

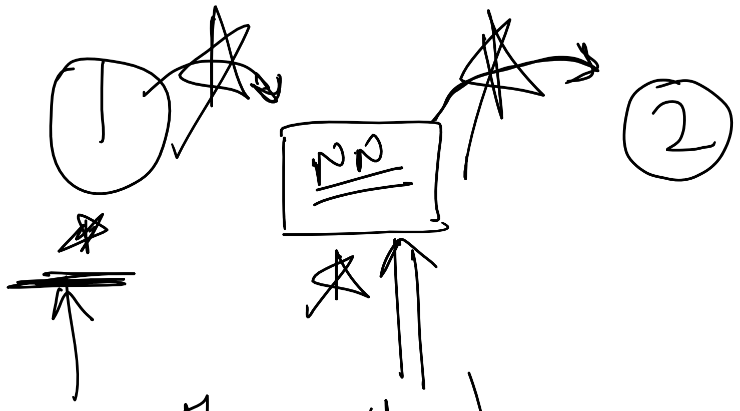


O/P



(1) 2nd position पर insert

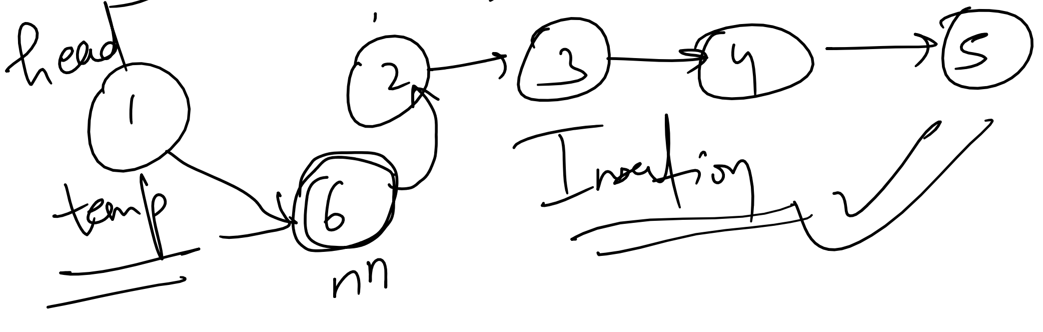




(nth position)

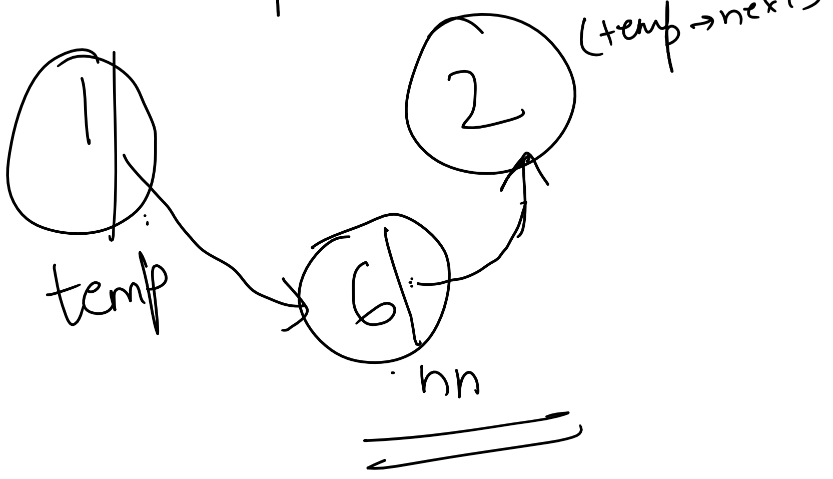
(n−1)th Node को Access   n=2
n−1 th Node

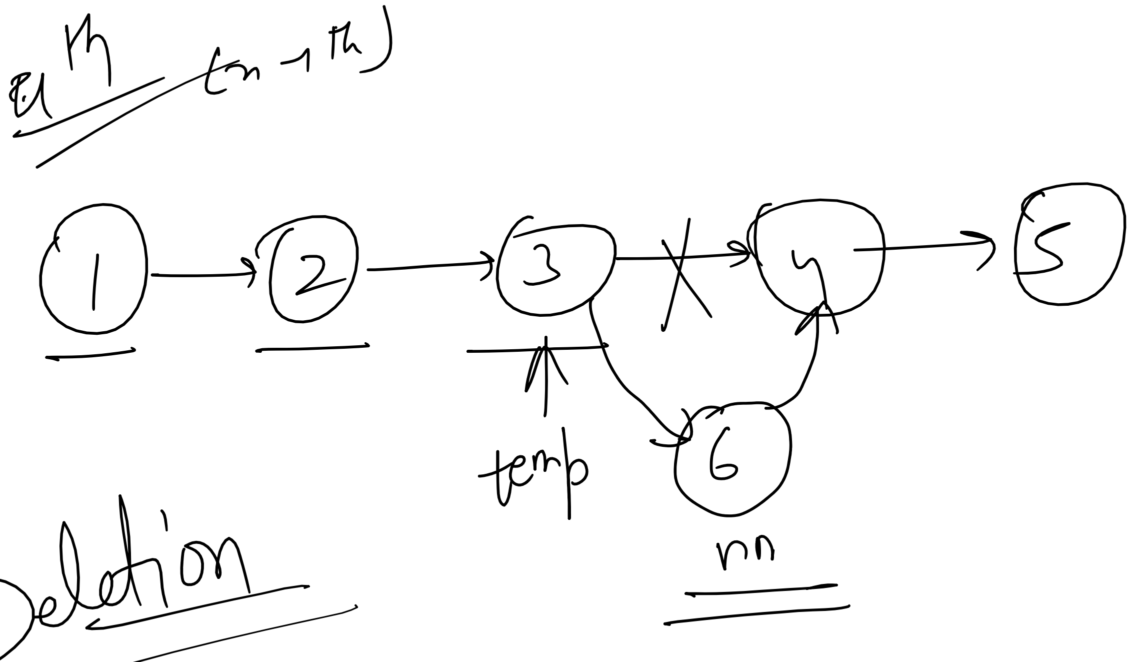
```
Node * temp = head;
i=1
while( i != (n-1)) {          i=1  (n-1) (1)
    temp = temp->next;
    i++;
}
```



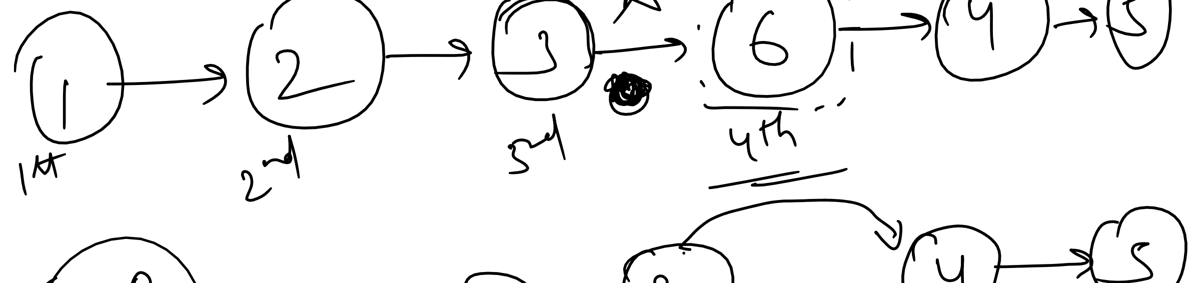
Insertion ✓

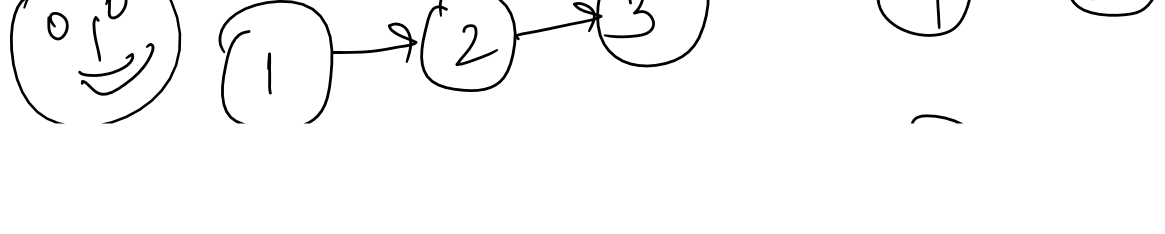
nn → next = temp → next
temp → next = nn;



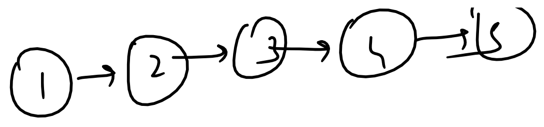


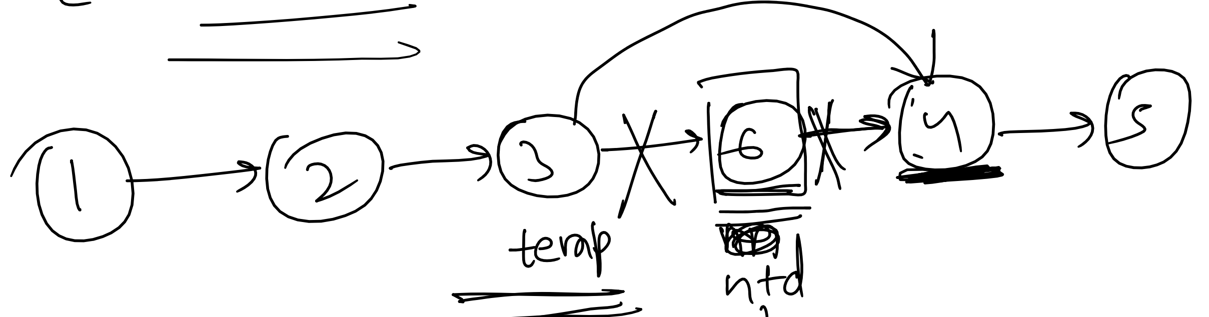
Deletion



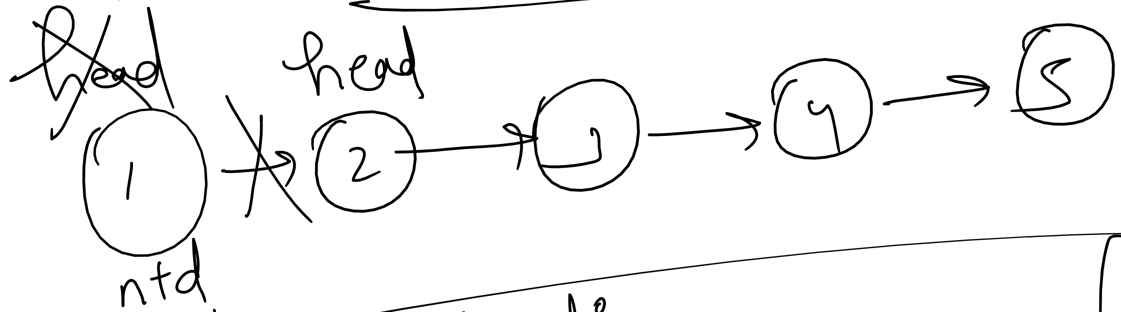


(n-1)th node

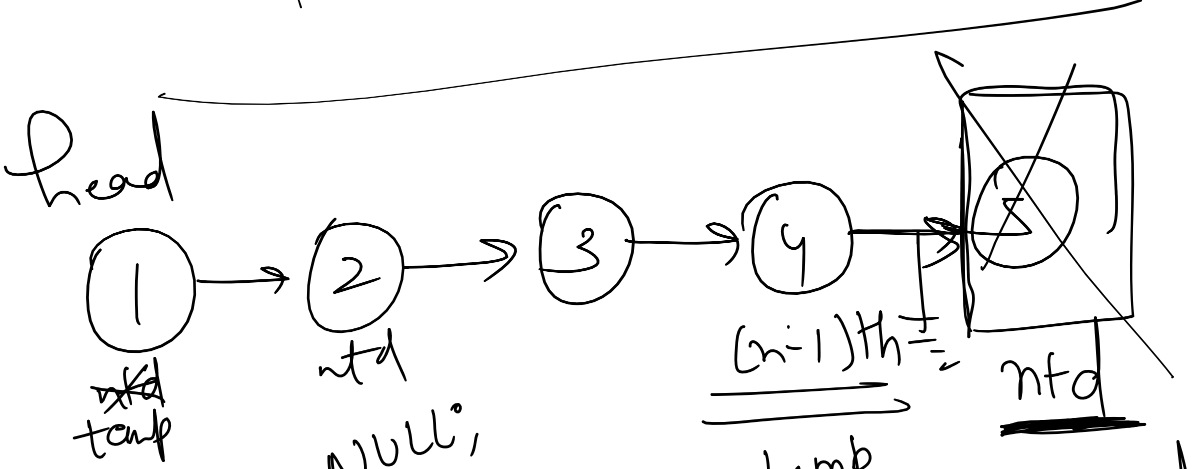




```
// ntd = temp→next
temp→next = ntd→next;
ntd→next = NULL
delete ntd;
```



```
ntd = head;
head = head→next;
ntd→next = NULL
delete ntd;
```



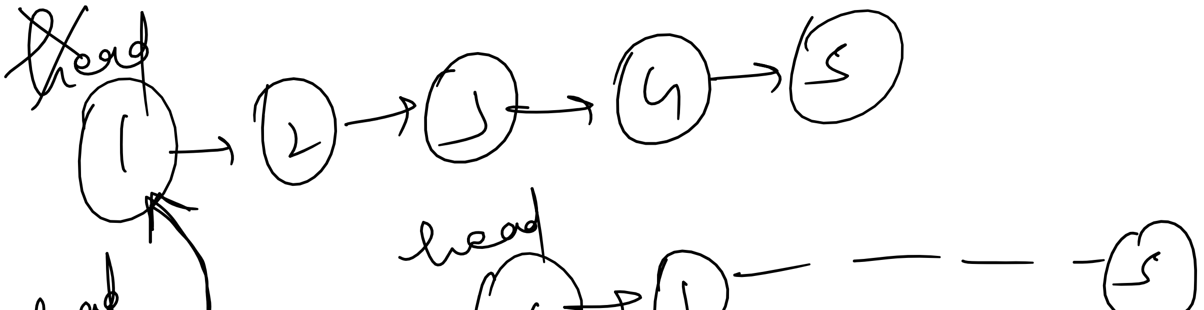
temp = NULL;

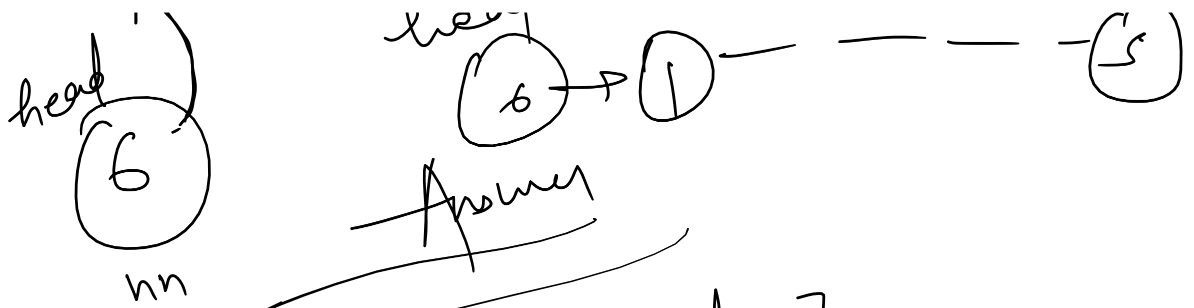
temp

```
temp = NULL;
ntd = head;
if( head == NULL)
{  X
}
else if (head->next == NULL)
{ head = NULL;   delete head;
}
else{
while (ntd->next != NULL)
{ X temp = ntd;
  ntd = ntd->next;
}
temp->next = NULL
delete ntd;
}
```

temp

1st insert

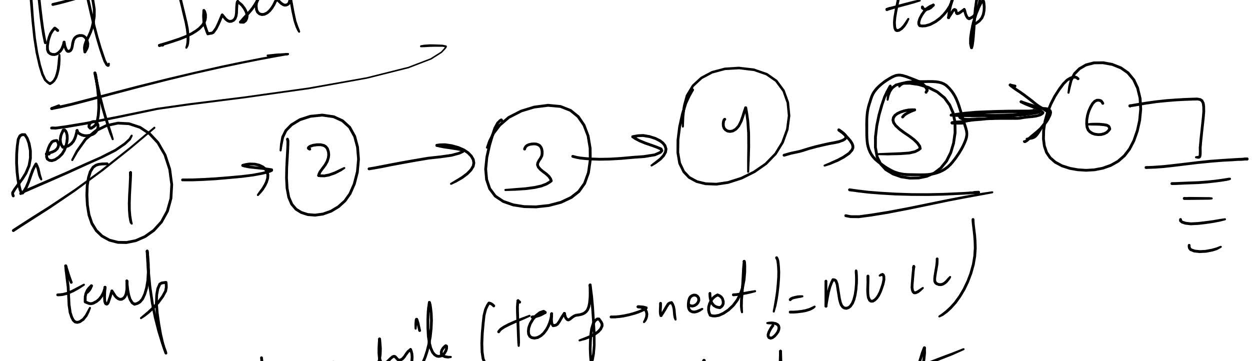




Answer

```
nn->next = head;
head = nn;
```

Last Insert



```
while (temp->next != NULL)
    { temp = temp->next
    }
temp->next = nn;
nn->next = NULL;
```