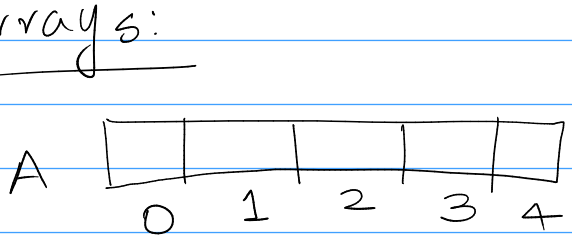
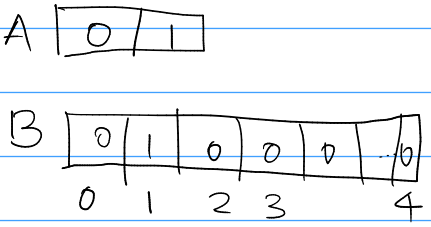
Arrays:

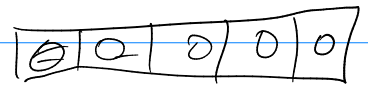


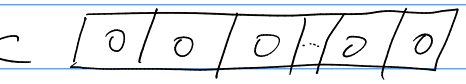


int A[] = {0, 1}; ✓ X ✓ int A[2] = {0, 1};

int B[10] = {0, 1}; ✓

int C[10] = {0}; X

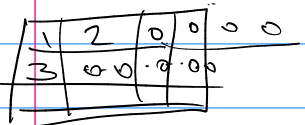




C[10] = {0}

---

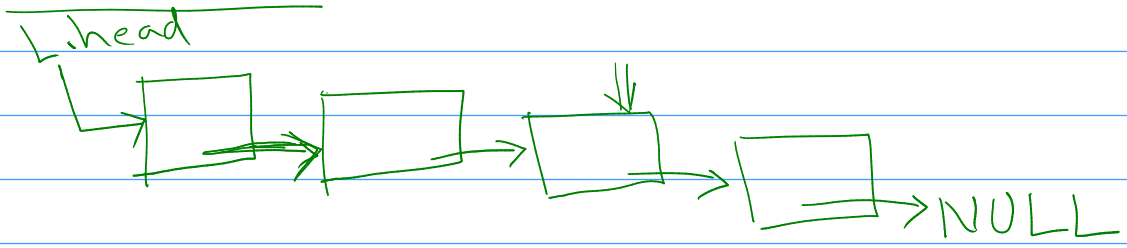
int P[10][10] = {[0][0] [0][1] [0][2] ... [0][9] [1][0] ... 1, 2, 3 ... 100}



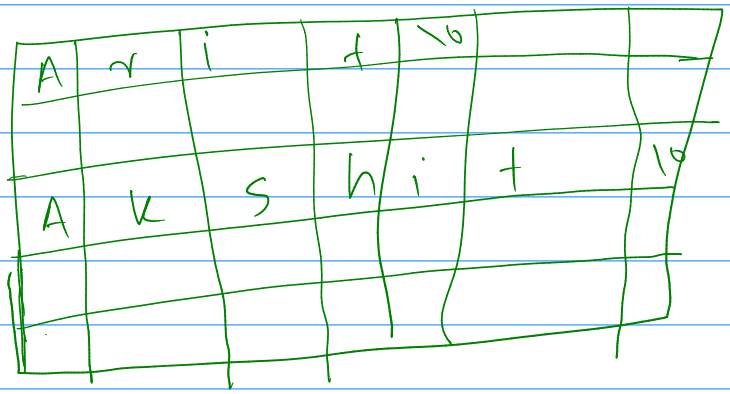
P[10][10] = {{1, 2}, {3},

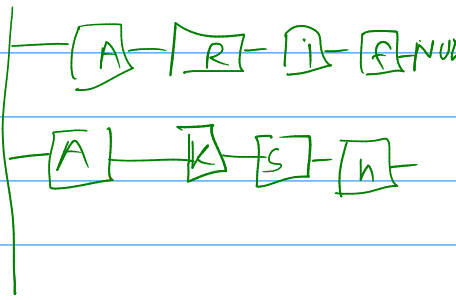
```
struct node {
    long int key;
    char name[10];
    struct node * next;
};
struct node p = {100, {'A','R',3}, NULL};
```

Linked List:

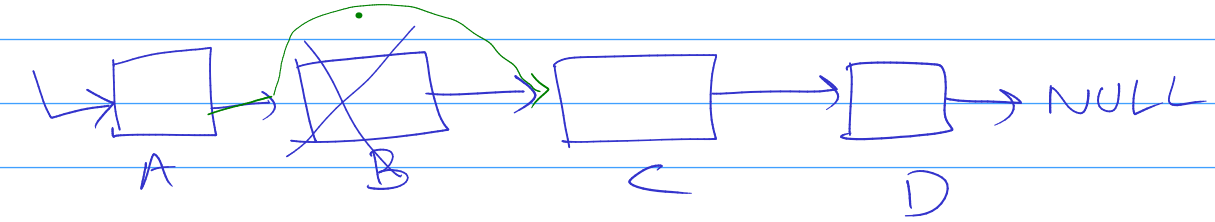


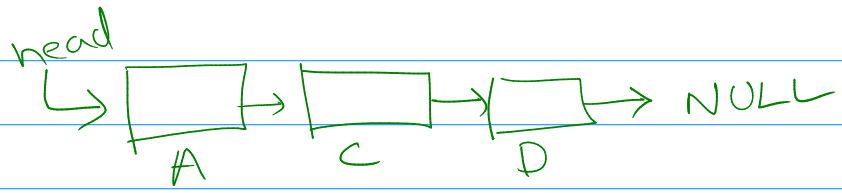

```
struct node {
    data;   =>  int key;
                char name[];

    Pointer variable of same type;
};
```





Delete an element from linked list:

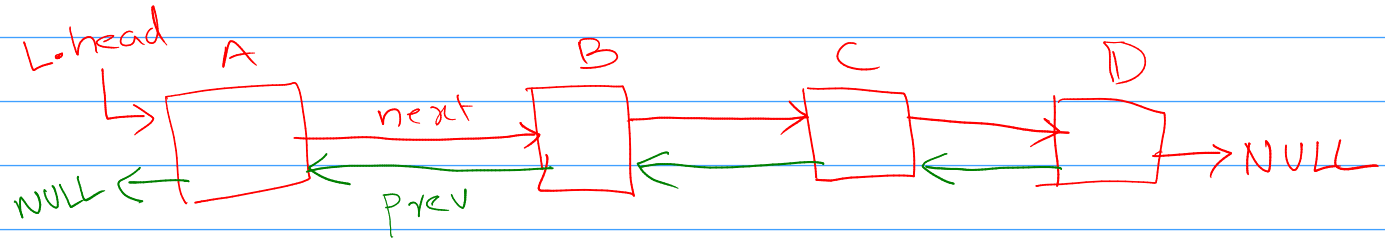




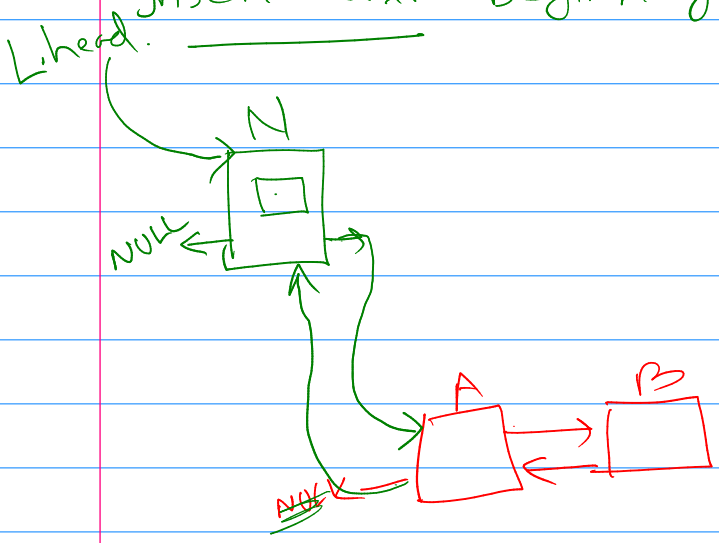

```
A->next = B->next;
free(B);
```

Oct 7, 2022

Doubly Linked List



Insert at the beginning:



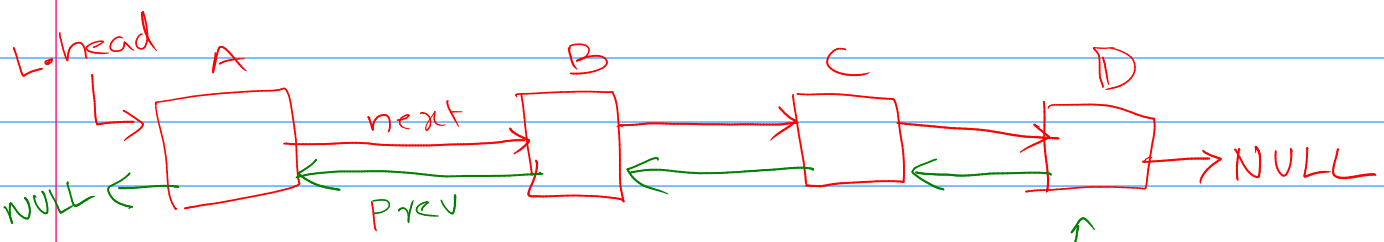
① N->next = L->head;

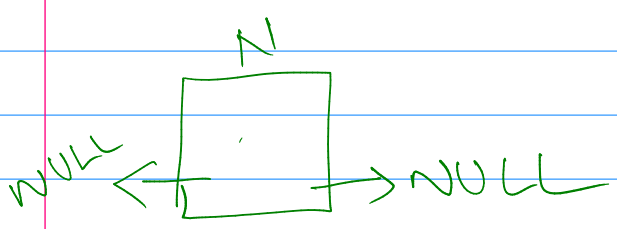
② L->head = N;

A->prev = N;

③ N->next->prev = N;

Insert at the End:



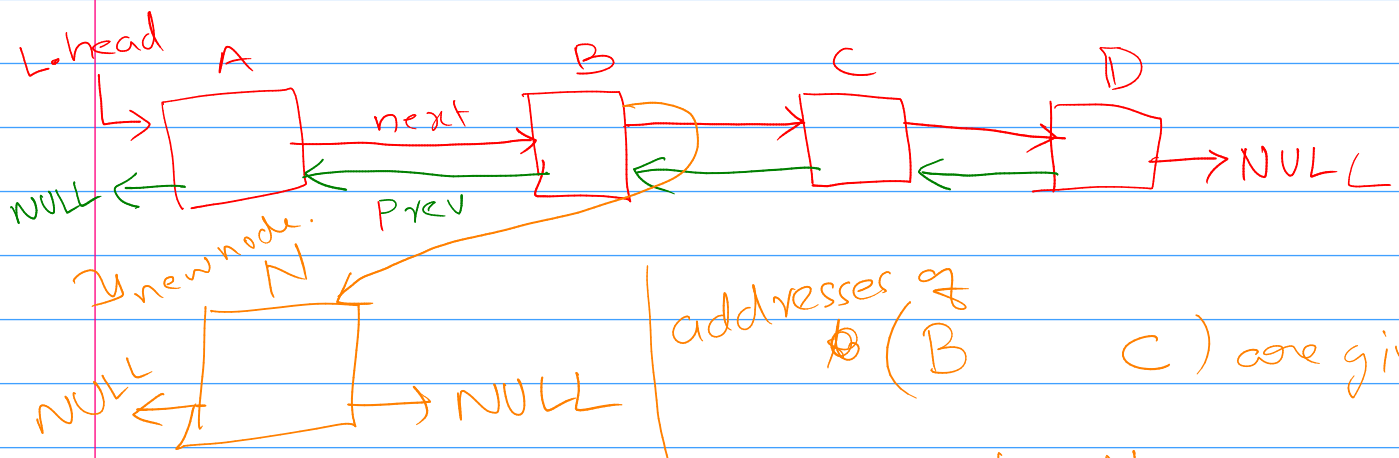


① Traverse upto

② temp->next = N;

③ N->prev = temp;

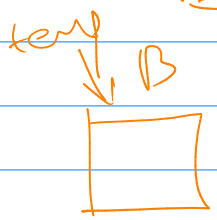
③ Insert in between: (between B and C);



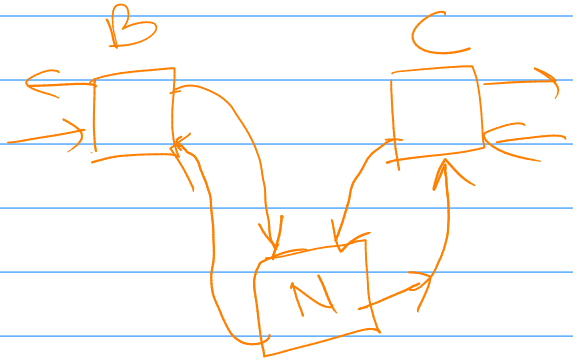
addresses of (B C) are given

① B → next = N
② N → next = C
③ C → prev = N
④ N → prev = B

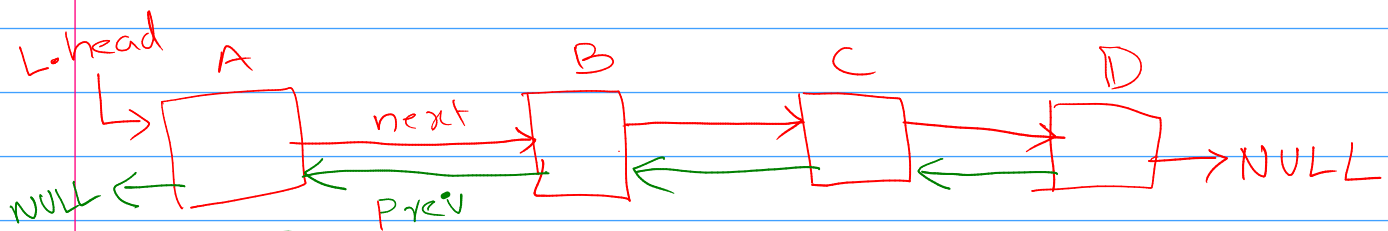
temp = head;
temp = temp → next;
temp = B;



① B → next → prev = N
② N → next = B → next;
③ B → next = N
④ N → prev = B



Delete the first element:



L->head

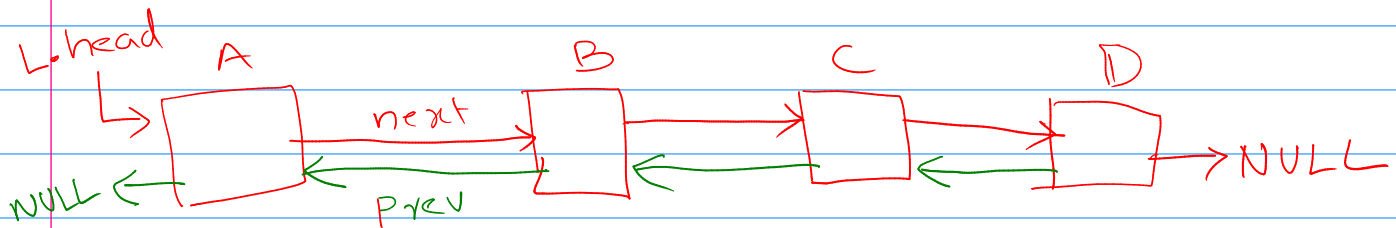
① ~~B=L->head~~ A=L->head; B=A->next;

② B->prev=NULL;

③ L->head = B;

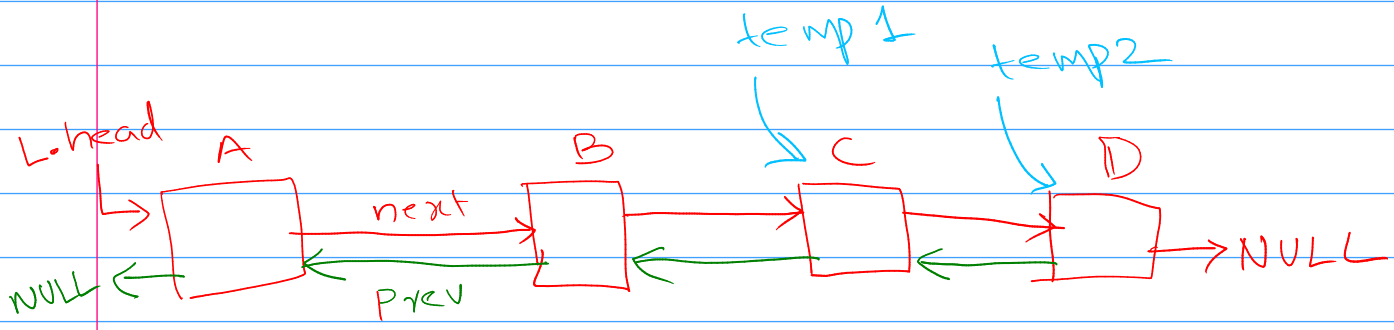
④ free(A);

Delete the last element:



L->head:

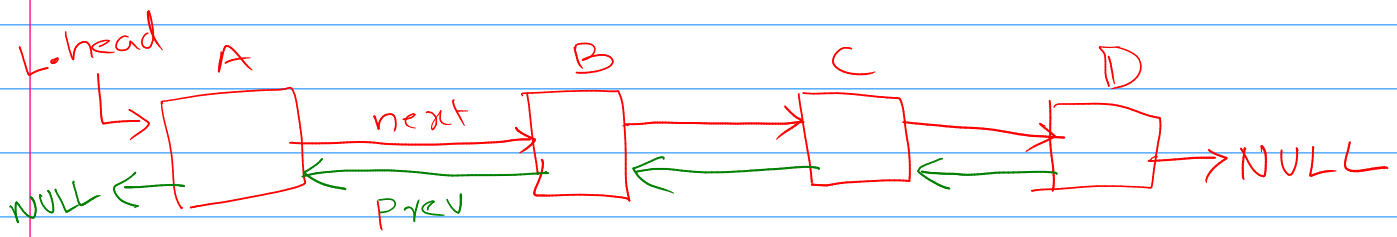
```
temp1= L->head;
temp2= L->head->next;
while( temp2->next != NULL) {
        temp1 = temp2;
        temp2 = temp2->next;
}
```



① temp1 → next = NULL;

② free(temp2);

Deleting in between: (Delete B);



L->head; (B)

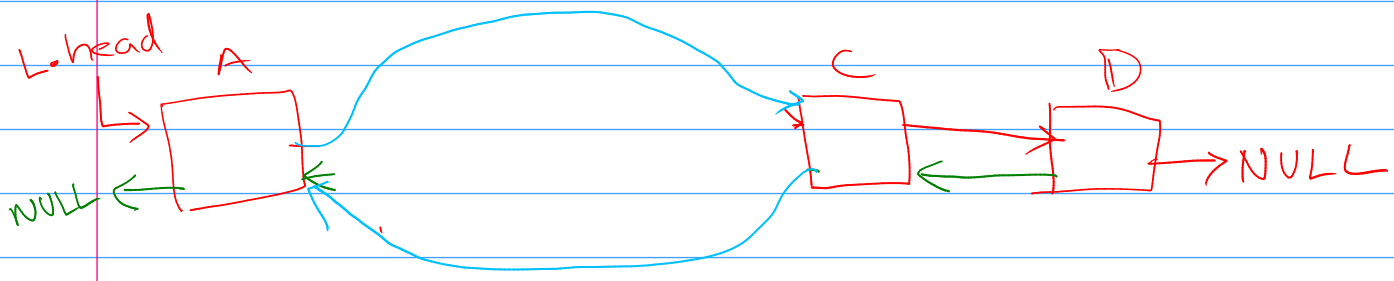
C = B → next;

A = B → prev;

① A → next = C;

② C → prev = A;

③ free(B);



L → head;

temp → (B)

① B → prev ⇒ next = C

② B → next → prev = A

③ free (B);