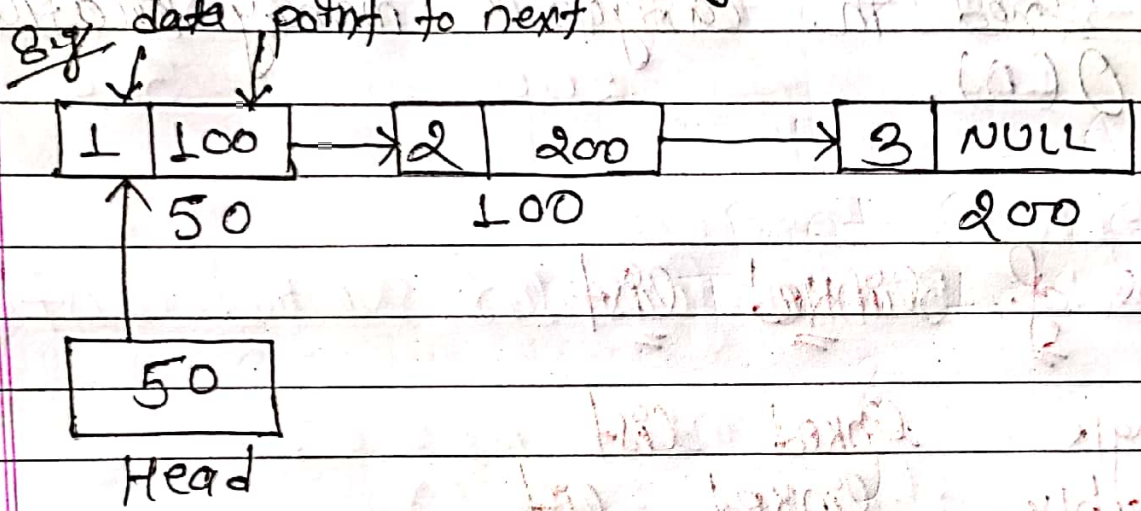
Linked list

→ In linked list elements are stored in non contiguous memory allocations.



→ In Array elements are stored in contiguous memory allocation.

eg

7	10	11	12	18	22

Why linked list?

→ memory & the capacity of an array remain fixed. In case of linked lists, we can keep adding & removing elements without any capacity constraints.

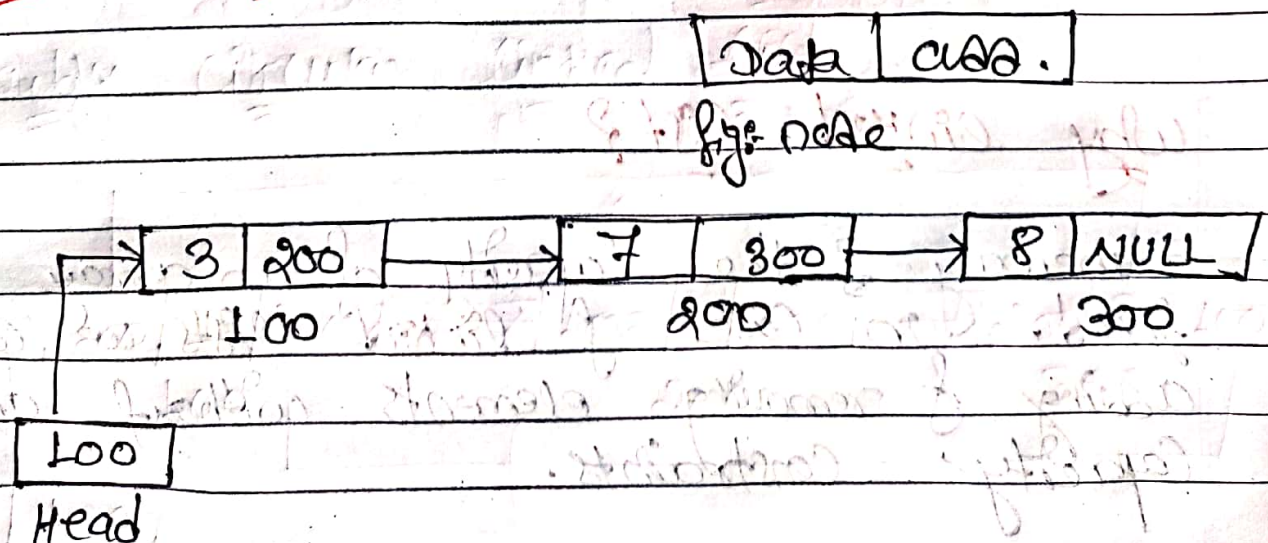
Drawbacks of linked lists

→ Extra memory space for pointers is required.
→ Random access not allowed as elements are not stored in contiguous memory locations. i.e. O(n).

Types of linked list

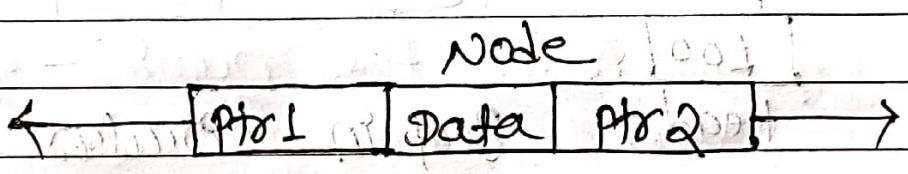
(i) Single linked list
(ii) Doubly linked list
(iii) Circular linked list
(iv) Doubly circular linked list

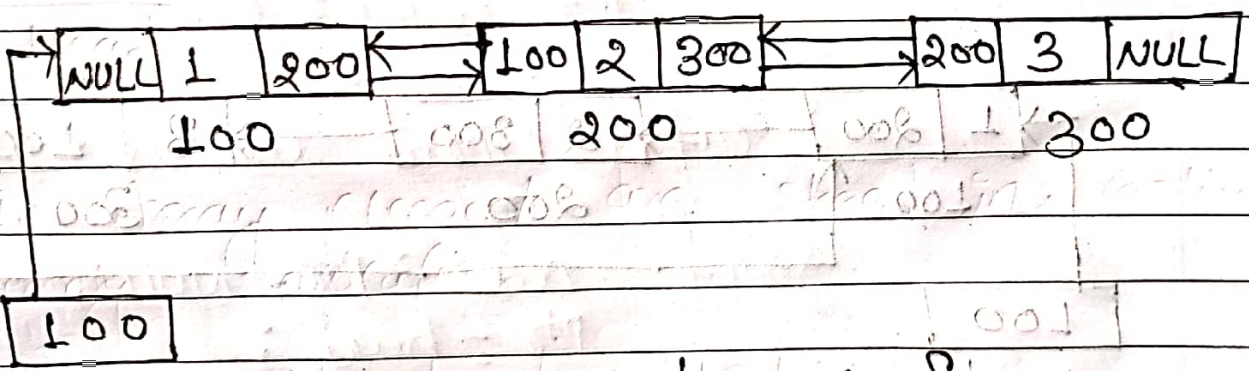
Single linked list



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

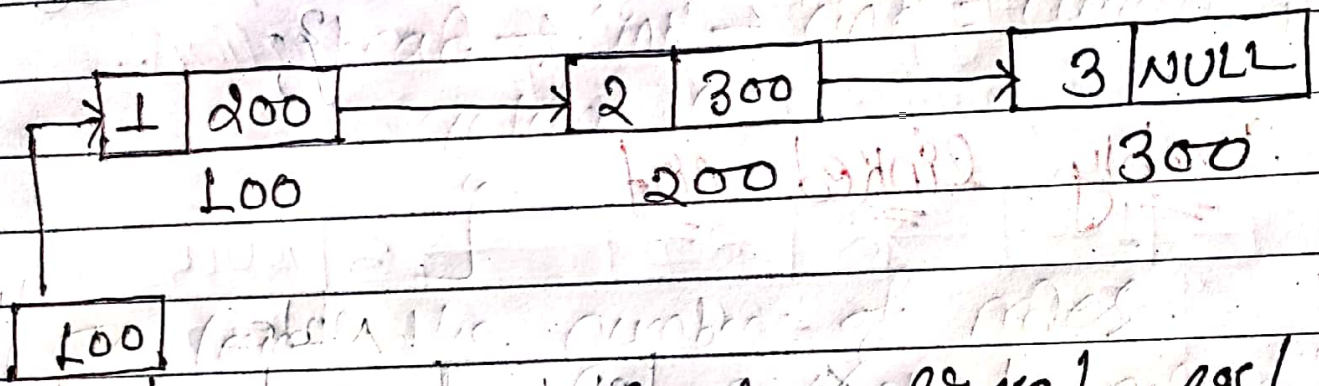
Doubly linked list



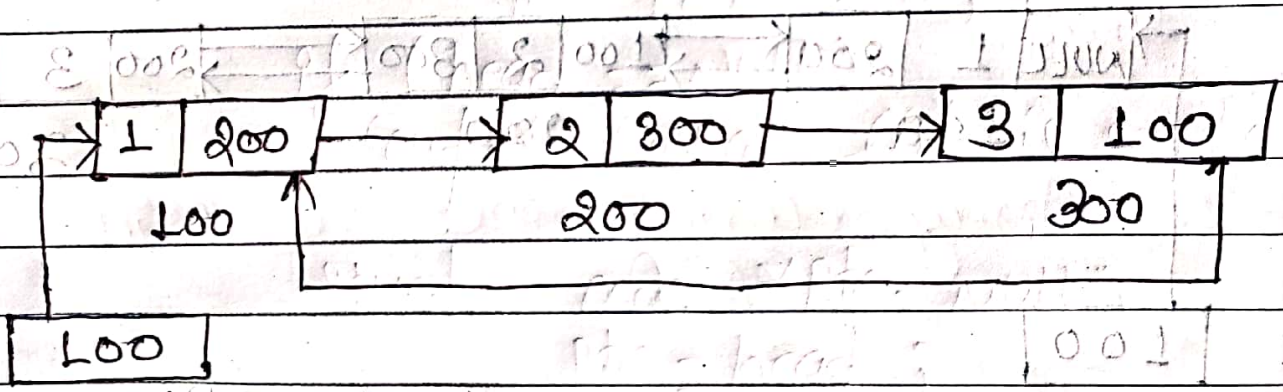



```
struct node {
    int data;
    struct node *next;
    struct node *prev;
};
```

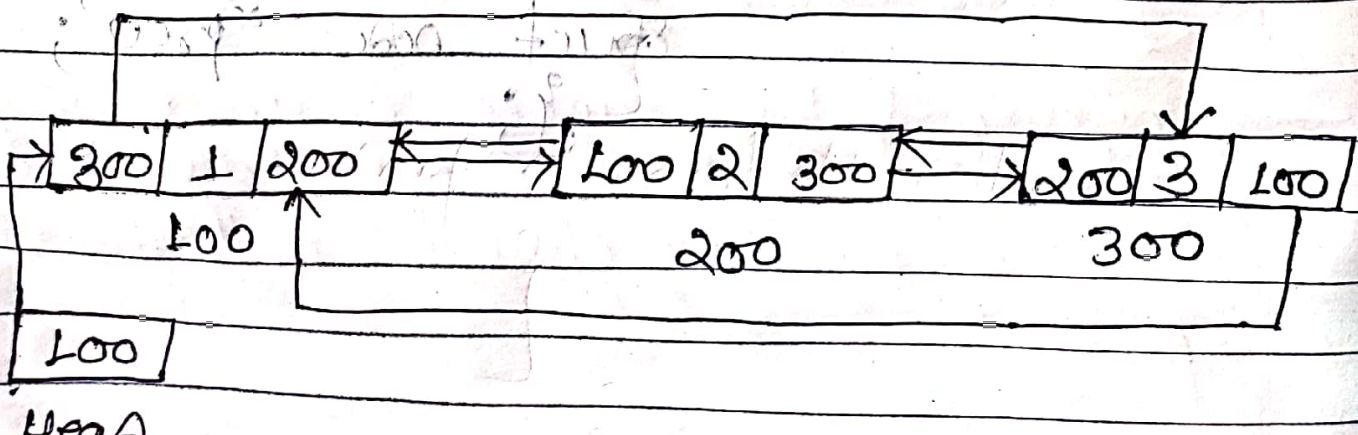
Single circular linked list / circular linked list



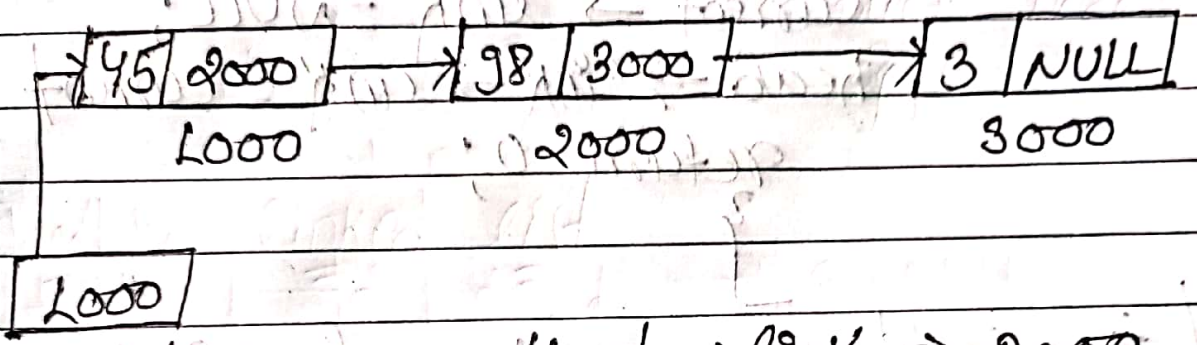
But in circular linked list



Doubly circular linked list



3 node



Head → link ⇒ 2000
Head → link → link ⇒ 3000
Head → link → link → link ⇒ NULL

```
class node {
    int data;
    node *link; // Self referencing structure
};

int main() {
    node *head = new node();
    head->data = 45;
    head->link = NULL;

    node *current = new node();
    current->data = 98;
    current->link = NULL;
    head->link = current;
```

```
current = new node ();
current -> data = 3;
current -> link = NULL;
head -> link -> link = current;
return 0;
}

// To count the number of nodes
void count_of_nodes(struct node *head){
    int count = 0;
    if (head == NULL)
        cout << "linked list is empty";
    struct node *ptr = NULL;
    ptr = head;

    while (ptr != NULL){
        count ++;
        ptr = ptr -> link;
    }
    cout << count << endl;
}
```

// Traversing a linked list.

```
void print_data (struct node *head ){
    if (head == NULL)
    cout << "linked list is empty";

struct node *ptr = NULL;
    ptr = head;
while (ptr != NULL){
    cout << ptr->data;
    ptr = ptr->link;
}
}
```

op :- 45 98 3

// To allocate memory dynamically

In C

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

In C++

```
head = new node ();
```