BIRZEIT UNIVERSITY

Faculty of Engineering and Technology

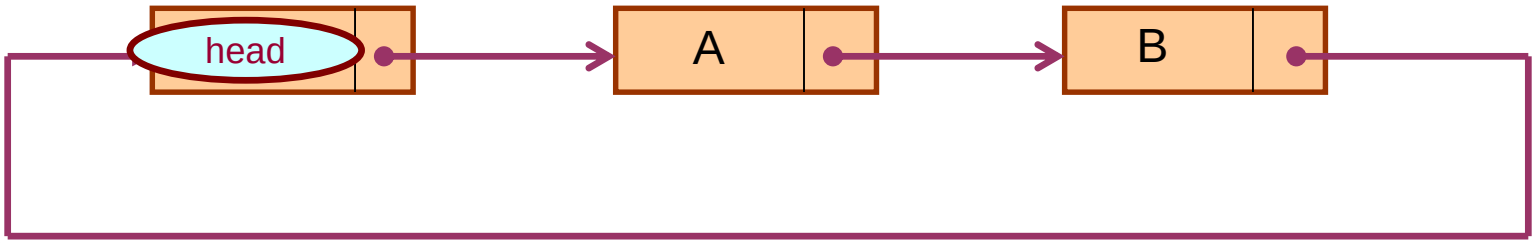
Computer Science Department

Linked Lists 2

- Circular Linked List
- Doubly Linked List
- Doubly Circular Linked List

Circular linked list

- Circular linked list
  - The pointer from the last element in the list points back to the first element.
  - How to recognize the head?



Operations in circular LL

- Creation

```
Linked_List L;
L=(Linked_List) malloc(sizeof(struct node));
L->next=L;
```

- Insertion

```
newNode->next=p->next;
p->next=newNode;
```

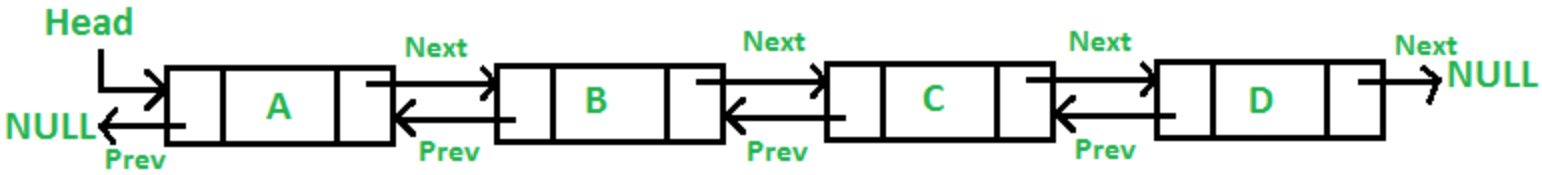
- visiting nodes

```
void Display_List_circular(Linked_List L)
{
    pos p = L->next;
    while(p != L)
    {
        printf("%d \n", p->ID);
        p=p->next;
    }

}
```

Doubly linked list

- Doubly linked list
  - Pointers exist between adjacent nodes in both directions.
  - Two pointers are maintained to keep track of the list, *next* and *previous*.
  - The list can be traversed either forward or backward.



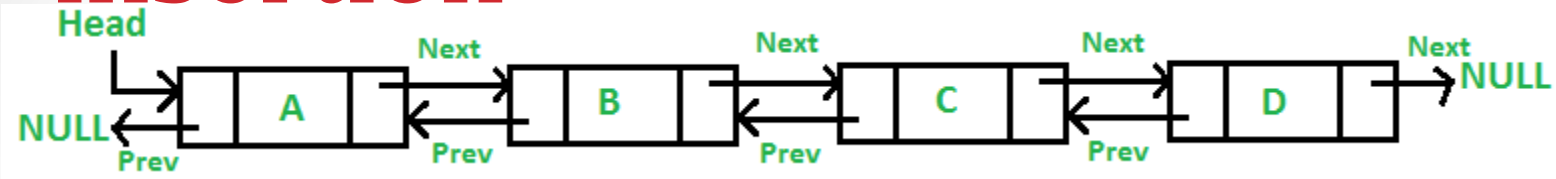

```
struct node;
typedef struct node  *ptr;
struct node
{
   int ID;
   ptr next;
   ptr prev;
};
```

creation

- Linked_List L;

- L=(Linked_List) malloc(sizeof(struct node));
- L->next=NULL;
- L->prev=NULL;

insertion



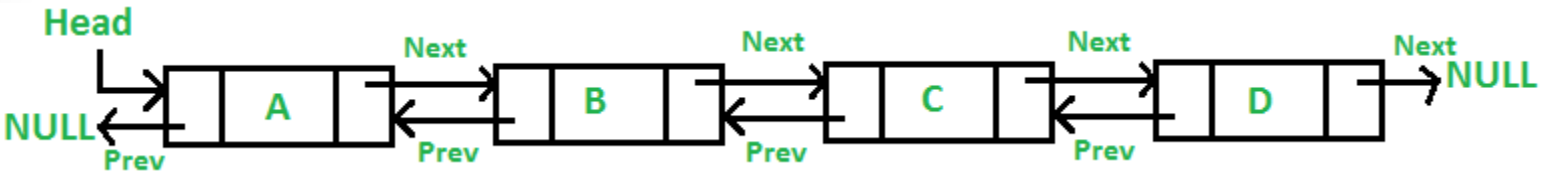

```
void insert(Linked_List L, pos p, ptr newNode)
{

   if(newNode !=NULL && L != NULL)
   {
      newNode->next=p->next;
      p->next=newNode;

      newNode->prev=p;
      if(newNode->next != NULL)
         newNode->next->prev=newNode;

      printf("Node # %d is inserted \n", newNode->ID);
   }
   else
      printf("ERROR either Linked List L or the new node is NULL \n");
}
```

Deletion




```
void delete_node(Linked_List L, ptr old)
{

   if(old!=NULL)
   {
      old->prev->next=old->next;
      old->next->prev=old->prev;
      free(old);
   }
}
```

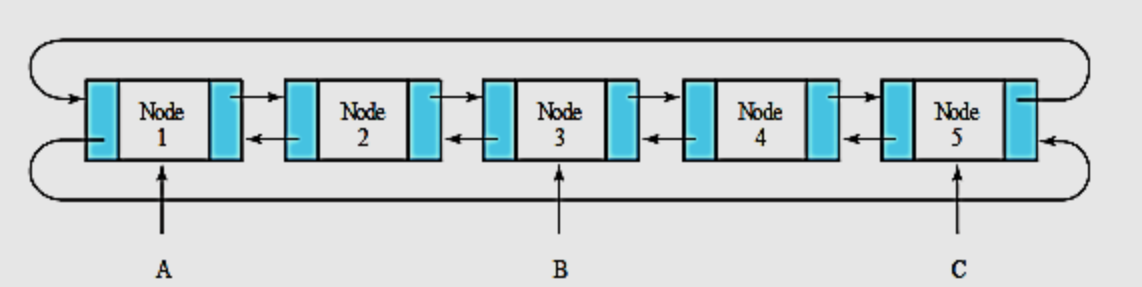
Be careful:

- Delete last node
- Delete the head

Other operations on DLL

- Visit all nodes
- Search
- Sort
- isEmpty

Circular doubly linked list



Questions:

Q1. Design a recursive algorithm that takes two sorted linked lists, and returns true if the lists are disjoint, i.e. have no elements in common. Your algorithm should take O(n) time.

Q2. Write a algorithm to reverse a given Doubly Linked List

Q3. Delete all occurrences of a given key in a doubly linked list

Given a doubly linked list and a key x. The problem is to delete all occurrences of the given key x from the doubly linked list.
Examples:

```
Input : DLL: 2 <-> 2 <-> 10 <-> 8 <-> 4 <-> 2 <-> 5 <-> 2
        x = 2
Output : 10 <-> 8 <-> 4 <-> 5
```

Q4. Remove duplicates from a sorted doubly linked list

Given a sorted doubly linked list containing n nodes. The problem is to remove duplicate nodes from the given
Examples:

```
Input : DLL: 4<->4<->4<->4<->6<->8<->8<->10<->12<->12
Output : 4<->6<->8<->10<->12
```

Q5. What is the average time, worst case, best case for (big-O):

1. Insert (at first, after element x, at end)
2. Delete (from first, element x, from end)

For

1. Single linked list
2. Double linked list
3. Circular single liked list
4. Circular Double liked list

Q6. Pairwise swap elements of a given linked list

Given a singly linked list, write a function to swap elements pairwise.

For example, if the linked list is 1->2->3->4->5 then the function should change it to 2->1->4->3->5, and if the linked list is 1->2->3->4->5->6 then the function should change it to 2->1->4->3->6->5.

Q.7 Given a singly linked list of characters, write a function that returns true if the given list is palindrome, else false.