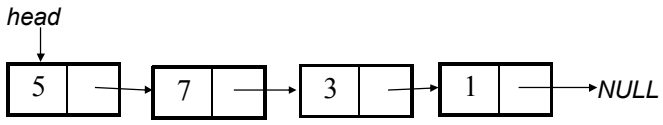
CSE 203: Data Structures and Algorithms-I

Linked Lists

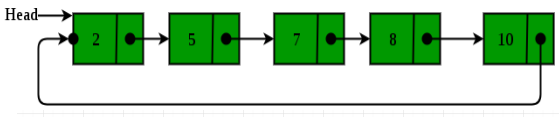


Dr. Md. Abul Kashem Mia, Professor, CSE Dept and Pro-Vice Chancellor, UIU

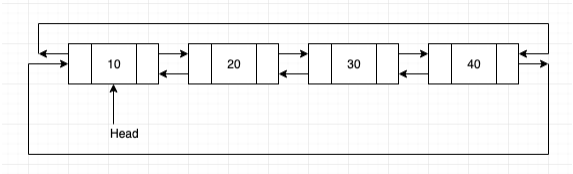
Circular Linked List

- In a circular linked list every element has a link to its next element and the last element has a link to the first element.
- That means circular linked list is similar to the single linked list except that the last node points to the first node in the list. There is no NULL at the end.
- A circular linked list can be a singly circular linked list or doubly circular linked list.

Singly circular linked list



Doubly circular linked list



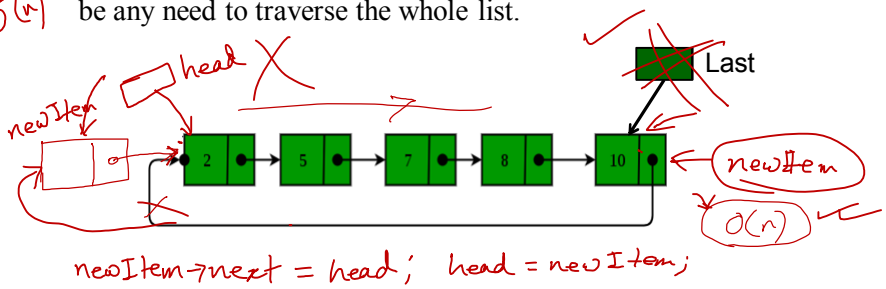
Dr. Md. Abul Kashem Mia, Professor, CSE Dept and Pro-Vice Chancellor, UIU

Singly Circular Linked List

- To implement a circular singly linked list, we take a global pointer that points to the last node instead of the Head node.

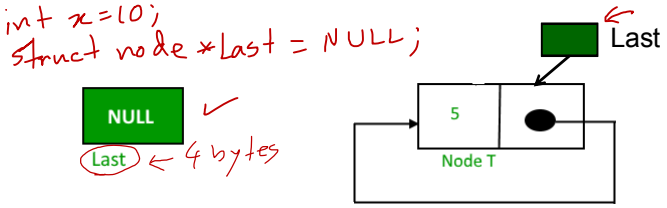
Why ?

- For the insertion at the beginning, the whole list has to be traversed.
- Also, for insertion at the end, the whole list has to be traversed.
- If we take a pointer to the last node, then in both cases there won't be any need to traverse the whole list.



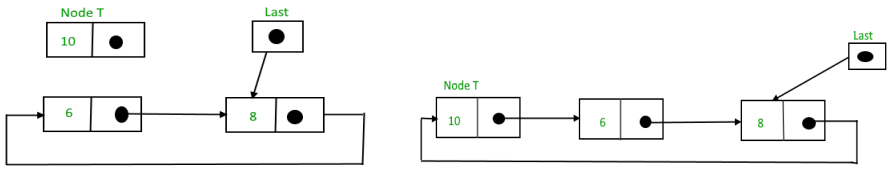
Circular SLL: Insertion in an Empty List

- Initially, when the list is empty, the last pointer will be NULL.




```
void  addToEmpty(int data){
   // Create a node dynamically
   struct node *newItem = (struct node*)malloc(sizeof(struct node));
    // Assign the data.
   newItem -> value = data;
   // Note : list was empty. We link single node to itself.
   last = newItem;
   newItem -> next = last;
}
```

Circular SLL: Insertion at the Beginning

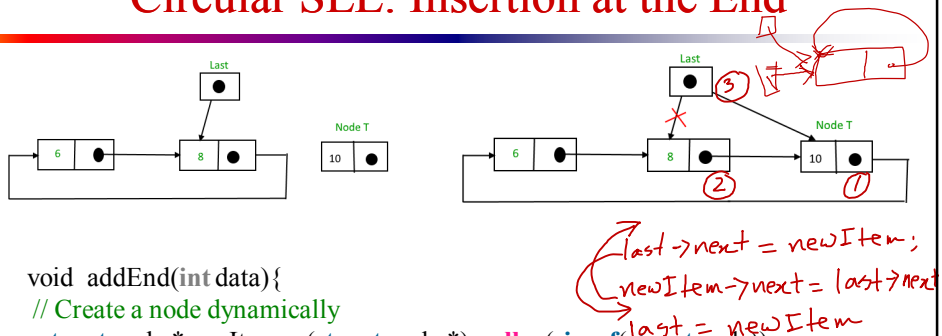



```
void  addBegin(int data){
 // Create a node dynamically
 struct node *newItem = (struct node *)malloc(sizeof(struct node));

 // Assign the data.
 newItem -> value = data;

 // Adjust the links.
 newItem -> next = last -> next;
 last -> next = newItem;
 }
```


Circular SLL: Insertion at the End




```
void  addEnd(int data){
 // Create a node dynamically
 struct node *newItem = (struct node *)malloc(sizeof(struct node));

 // Assign the data
  newItem -> value = data;

 // Adjust the links
 newItem -> next = last -> next;
 last -> next = newItem;
 last = newItem;
 }
```

Summary on Linked Lists

- **Advantages:**
  - Linked List is dynamic data structure, that is, the Linked List grows dynamically.
  - Insertion into Linked Lists and deletion from Linked Lists are very fast ($O(1)$ time).
- **Disadvantages:**
  - Linked List is a sequential access data structure.
  - Accessing an element by pointers is very slow ($O(n)$ time)
- A Linked List is a suitable structure when
  - a lot of insertions and deletions are required.
  - a small number of searching and retrieval are required.