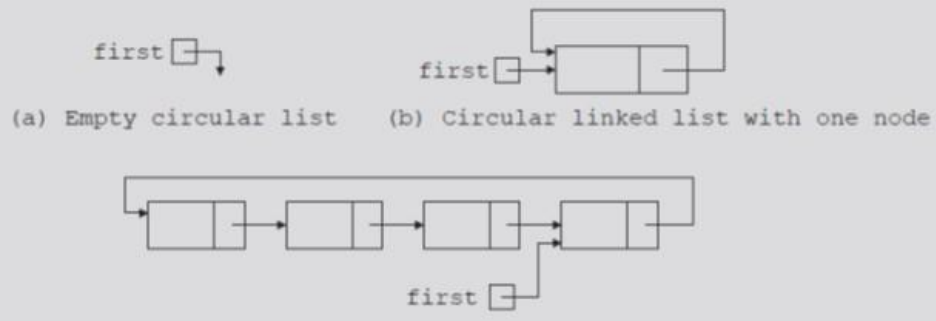
The School Electrical Engineering and Information Technology
Computer Science Department

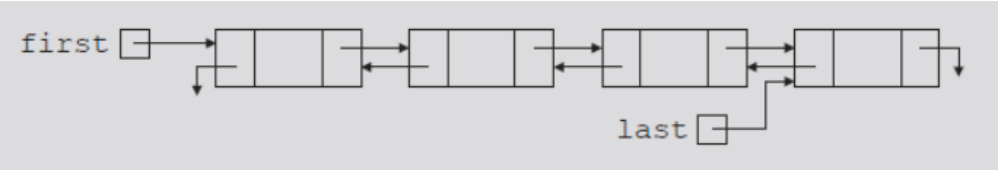
CS223 Lab
Circular Linked List
Doubly linked List

Definition:

- A circular linked list is a variation of the linked list. It is a linked list whose nodes are connected in such a way that it forms a circle. , Eg, The following is a linked list with 4 nodes:



- Doubly Linked list: is a list in which every node contains address of next node except last node and every node contains address of previous node, Eg, The following is a linked list with 4 nodes:



A structure, which contains a data element and pointers to the next and previous nodes, is created as follows:

```
struct Node {

int value;

struct Node *next;
struct Node *previous;
 };
```

Mainly we use pointers to be a connector between nodes. Every time we need to add a new node, we uses the connectors to add it. This is why using the keyword new to create a pointer to a new struct.

Main Operations

• Insertion
Adding a new node to the linked list in the beginning, end, or in between two nodes and adjusting the necessary connections.

• Deletion
Removing a given node from the linked list and adjusting the necessary connections.

• Traversal
Visiting each node once for doing something to that node, such as displaying the data in the node.

• Searching
Finding the location of a given item of information in the linked list and returning a pointer to the node.

Lab Work

Consider the following C++ code, which contains the full implementation of a doubly linked list with thefollowing operations:

- Define a structure node
- Declare Head node
- **Insertfirst** function to add a node at the beginning of the linked list.
- Traverse the linked list to print it

```
#include <iostream>
using namespace std;
   struct Node {
   int data;
    Node *next; // Pointer to next node
    Node *prev; // Pointer to previous node
};
//Declare Head node
  struct Node *head = NULL;
   //insert at first
   void insertfirst( int n)
{
   Node *new_node = new Node;
   new_node->data = n;
   new_node->next = head;
   new_node->prev = NULL;
      if (head != NULL)
     head->prev = new_node;
     head = new_node;
}
```

```
  //insert last
void deletefirst()
  {

  }
// void duplicate()
{

}
//Tavesrse
void traverse(){
   if(head==NULL){
      cout<<"Empty List!"<<endl;
      return;
   }
    Node *temp=head;
   while(temp!=NULL){
      cout<<temp->data<<" ";
      temp=temp->next;
}   }
int main()
{
insertfirst(1);
insertfirst(2);
insertfirst(3);
insertfirst(4);
insertfirst(5);
traverse();
   return 0;
}
```

Lab Exercises:

1. Compile and run the previous code.

2. Fill in the blank the following two functions:
    - Delete function to delete the first node.
    - Duplicate function to duplicate the last node :

If the list is: 1 2 3 4 5
It will be: 1 2 3 4 5 5