Traverse a Doubly Linked List

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  public:
    Node* head;
  public:
    //constructor to create an empty LinkedList
    LinkedList(){
      head = NULL;
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  //create an empty LinkedList
  LinkedList MyList;

  //Add first node.
  Node* first = new Node();
  first->data = 10;
  first->next = NULL;
  first->prev = NULL;
```

```
  //linking with head node
  MyList.head = first;

  //Add second node.
  Node* second = new Node();
  second->data = 20;
  second->next = NULL;
  //linking with first node
  second->prev = first;
  first->next = second;

  //Add third node.
  Node* third = new Node();
  third->data = 30;
  third->next = NULL;
  //linking with second node
  third->prev = second;
  second->next = third;

  //print the content of list
  MyList.PrintList();
  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30
```

Doubly Linked List - Insert a new node at the start

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
```

```
      head = NULL;
    }

    //Add new element at the start of the list
    void push_front(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        head->prev = newNode;
        newNode->next = head;
        head = newNode;
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add three elements at the start of the list.
  MyList.push_front(10);
  MyList.push_front(20);
  MyList.push_front(30);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 30 20 10
```

Doubly Linked List - Insert a new node at the end

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
```

```
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add three elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30
```

Doubly Linked List - Insert a new node at the given position

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
```

```
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Inserts a new element at the given position
    void push_at(int newElement, int position) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(position < 1) {
        cout<<"\nposition should be >= 1.";
      } else if (position == 1) {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
      } else {
        Node* temp = head;
        for(int i = 1; i < position-1; i++) {
          if(temp != NULL) {
            temp = temp->next;
          }
        }
        if(temp != NULL) {
          newNode->next = temp->next;
          newNode->prev = temp;
          temp->next = newNode;
          if(newNode->next != NULL)
            newNode->next->prev = newNode;
        } else {
          cout<<"\nThe previous node is null.";
        }
      }
```

```
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add three elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.PrintList();

  //Insert an element at position 2
  MyList.push_at(100, 2);
  MyList.PrintList();

  //Insert an element at position 1
  MyList.push_at(200, 1);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30
The list contains: 10 100 20 30
The list contains: 200 10 100 20 30
```

Doubly Linked List - Delete the first node

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Delete first node of the list
    void pop_front() {
      if(head != NULL) {
        Node* temp = head;
        head = head->next;
        free(temp);
        if(head != NULL)
          head->prev = NULL;
      }
```

```
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add four elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.PrintList();

  //Delete the first node
  MyList.pop_front();
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40
The list contains: 20 30 40
```

Doubly Linked List - Delete the last node

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Delete last node of the list
    void pop_back() {
      if(head != NULL) {
        if(head->next == NULL) {
          head = NULL;
        } else {
          Node* temp = head;
          while(temp->next->next != NULL)
            temp = temp->next;
```

```
          Node* lastNode = temp->next;
          temp->next = NULL;
          free(lastNode);
        }
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add four elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.PrintList();

  //Delete the last node
  MyList.pop_back();
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40
The list contains: 10 20 30
```

Doubly Linked List - Delete a node at the given position

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Delete an element at the given position
    void pop_at(int position) {
      if(position < 1) {
        cout<<"\nposition should be >= 1.";
      } else if (position == 1 && head != NULL) {
        Node* nodeToDelete = head;
        head = head->next;
        free(nodeToDelete);
        if(head != NULL)
          head->prev = NULL;
      } else {
```

```
        Node* temp = head;
        for(int i = 1; i < position-1; i++) {
          if(temp != NULL) {
            temp = temp->next;
          }
        }
        if(temp != NULL && temp->next != NULL) {
          Node* nodeToDelete = temp->next;
          temp->next = temp->next->next;
          if(temp->next->next != NULL)
            temp->next->next->prev = temp->next;
          free(nodeToDelete);
        } else {
          cout<<"\nThe node is already null.";
        }
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add three elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.PrintList();

  //Delete an element at position 2
  MyList.pop_at(2);
  MyList.PrintList();
```

```
  //Delete an element at position 1
  MyList.pop_at(1);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30
The list contains: 10 30
The list contains: 30
```

Doubly Linked List - Delete all nodes

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
```

```
        newNode->prev = temp;
      }
    }

    //delete all nodes of the list
    void deleteAllNodes() {
      Node* temp = new Node();
      while(head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
      }
      cout<<"All nodes are deleted successfully.\n";
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add four elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);

  //Display the content of the list.
  MyList.PrintList();

  //delete all nodes of the list
  MyList.deleteAllNodes();

  //Display the content of the list.
```

```
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40
All nodes are deleted successfully.
The list is empty.
```

Doubly Linked List - Count nodes

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
```

```
    }

    //count nodes in the list
    int countNodes() {
      Node* temp = head;
      int i = 0;
      while(temp != NULL) {
        i++;
        temp = temp->next;
      }
      return i;
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add four elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);

  //Display the content of the list.
  MyList.PrintList();

  //number of nodes in the list
  cout<<"No. of nodes: "<<MyList.countNodes();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40
No. of nodes: 4
```

Doubly Linked List - Delete even nodes

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //delete even nodes of the list
    void deleteEvenNodes() {
      if(head != NULL) {
```

```
        Node* oddNode = head;
        Node* evenNode = head->next;
        Node* temp = new Node();

        while(oddNode != NULL && evenNode != NULL) {
          oddNode->next = evenNode->next;
          free(evenNode);

          temp = oddNode;
          oddNode = oddNode->next;
          if(oddNode != NULL){
            oddNode->prev = temp;
            evenNode = oddNode->next;
          }
        }
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add five elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.push_back(50);

  //Display the content of the list.
  MyList.PrintList();
```

```
  //delete even nodes of the list
  MyList.deleteEvenNodes();

  cout<<"After deleting even nodes.\n";
  //Display the content of the list.
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40 50
After deleting even nodes.
The list contains: 10 30 50
```

Doubly Linked List - Delete odd nodes

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
```

```
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //delete odd nodes of the list
       void deleteOddNodes() {
         if(head != NULL) {
           Node* temp = head;
           head = head->next;
           free(temp);
           if(head != NULL) {
             head->prev = NULL;
             Node* evenNode = head;
             Node* oddNode = head->next;
             while(evenNode != NULL && oddNode != NULL) {
               evenNode->next = oddNode->next;
               free(oddNode);
               temp = evenNode;
               evenNode = evenNode->next;
               if(evenNode != NULL) {
                 evenNode->prev = temp;
                 oddNode = evenNode->next;
               }
             }
           }
         }
       }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
```

```
int main() {
  LinkedList MyList;

  //Add five elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.push_back(50);

  //Display the content of the list.
  MyList.PrintList();

  //delete odd nodes of the list
  MyList.deleteOddNodes();

  cout<<"After deleting odd nodes.\n";
  //Display the content of the list.
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40 50
After deleting odd nodes.
The list contains: 20 40
```

Doubly Linked List - Search an element

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
```

```
    head = NULL;
  }

  //Add new element at the end of the list
  void push_back(int newElement) {
    Node* newNode = new Node();
    newNode->data = newElement;
    newNode->next = NULL;
    newNode->prev = NULL;
    if(head == NULL) {
      head = newNode;
    } else {
      Node* temp = head;
      while(temp->next != NULL)
        temp = temp->next;
      temp->next = newNode;
      newNode->prev = temp;
    }
  }

  //Search an element in the list
  void SearchElement(int searchValue) {
    Node* temp = head;
    int found = 0;
    int i = 0;

    if(temp != NULL) {
      while(temp != NULL) {
        i++;
        if(temp->data == searchValue) {
          found++;
          break;
        }
        temp = temp->next;
      }
      if (found == 1) {
        cout<<searchValue<<" is found at index = "<<i<<".\n";
      } else {
        cout<<searchValue<<" is not found in the list.\n";
      }
    } else {
      cout<<"The list is empty.\n";
    }
  }

  //display the content of the list
  void PrintList() {
    Node* temp = head;
```

```
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add three elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);

  //traverse to display the content of the list.
  MyList.PrintList();

  //search for element in the list
  MyList.SearchElement(10);
  MyList.SearchElement(15);
  MyList.SearchElement(20);

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30
10 is found at index = 1.
15 is not found in the list.
20 is found at index = 2.
```

Doubly Linked List - Delete first node by key

```
#include <iostream>
using namespace std;
```

```
//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Delete first node by key
    void pop_first(int key) {
      Node* temp = head;
      if(temp != NULL) {
        if(temp->data == key) {
          Node* nodeToDelete = head;
          head = head->next;
          free(nodeToDelete);
          if(head != NULL)
            head->prev = NULL;
        } else {
          while(temp->next != NULL) {
            if(temp->next->data == key) {
              Node* nodeToDelete = temp->next;
              temp->next = temp->next->next;
              if(temp->next != NULL)
```

```
                temp->next->prev = temp;
              free(nodeToDelete);
              break;
            }
            temp = temp->next;
          }
        }
      }
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add five elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.PrintList();

  //Delete the first occurrence of 20
  MyList.pop_first(20);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 10 20
The list contains: 10 30 10 20
```

Doubly Linked List - Delete last node by key

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //Delete last node by key
    void pop_last(int key) {

      if(head != NULL) {
        Node *previousToLast, *lastNode, *temp;
        previousToLast = NULL;
```

```
        lastNode = NULL;

        if(head->data == key)
          lastNode = head;

        temp = head;
        while(temp->next != NULL) {
          if(temp->next->data == key) {
            previousToLast = temp;
            lastNode = temp->next;
          }
          temp = temp->next;
        }

        if(lastNode != NULL) {
          if(lastNode == head) {
            head = head->next;
            free(lastNode);
          } else {
            previousToLast->next = lastNode->next;
            if(previousToLast->next != NULL)
              previousToLast->next->prev = previousToLast;
            free(lastNode);
          }
        }
      }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;
```

```
  //Add five elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(20);
  MyList.push_back(40);
  MyList.PrintList();

  //Delete the last occurrence of 20
  MyList.pop_last(20);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 20 40
The list contains: 10 20 30 40
```

Doubly Linked List - Delete all nodes by key

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
```

```
    newNode->prev = NULL;
    if(head == NULL) {
      head = newNode;
    } else {
      Node* temp = head;
      while(temp->next != NULL)
        temp = temp->next;
      temp->next = newNode;
      newNode->prev = temp;
    }
  }

  //Delete all nodes by key
  void pop_all(int key) {
    Node* nodeToDelete;
    while(head != NULL && head->data == key) {
      nodeToDelete = head;
      head = head->next;
      free(nodeToDelete);
      if(head != NULL)
        head->prev = NULL;
    }

    Node* temp = head;
    if(temp != NULL) {
      while(temp->next != NULL) {
        if(temp->next->data == key) {
          nodeToDelete = temp->next;
          temp->next = temp->next->next;
          if(temp->next != NULL)
            temp->next->prev = temp;
          free(nodeToDelete);
        } else {
          temp = temp->next;
        }
      }
    }
  }

  //display the content of the list
  void PrintList() {
    Node* temp = head;
    if(temp != NULL) {
      cout<<"The list contains: ";
      while(temp != NULL) {
        cout<<temp->data<<" ";
        temp = temp->next;
      }
```

```
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add five elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.PrintList();

  //Delete all occurrences of 20
  MyList.pop_all(20);
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 10 20
The list contains: 10 30 10
```

Doubly Linked List - Reverse the List

```
#include <iostream>
using namespace std;

//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
```

```
public:
  LinkedList(){
    head = NULL;
  }

  //Add new element at the end of the list
  void push_back(int newElement) {
    Node* newNode = new Node();
    newNode->data = newElement;
    newNode->next = NULL;
    newNode->prev = NULL;
    if(head == NULL) {
      head = newNode;
    } else {
      Node* temp = head;
      while(temp->next != NULL)
        temp = temp->next;
      temp->next = newNode;
      newNode->prev = temp;
    }
  }

  //reverse the list
  void reverseList() {
    if(head != NULL) {
      Node* prevNode = head;
      Node* tempNode = head;
      Node* curNode = head->next;

      prevNode->next = NULL;
      prevNode->prev = NULL;

      while(curNode != NULL) {
        tempNode = curNode->next;
        curNode->next = prevNode;
        prevNode->prev = curNode;
        prevNode = curNode;
        curNode = tempNode;
      }

      head = prevNode;
    }
  }

  //display the content of the list
  void PrintList() {
    Node* temp = head;
    if(temp != NULL) {
```

```
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add five elements at the end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.push_back(50);

  //Display the content of the list.
  MyList.PrintList();

  //Reversing the list.
  MyList.reverseList();

  //Display the content of the list.
  MyList.PrintList();

  return 0;
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40 50
The list contains: 50 40 30 20 10
```

Doubly Linked List - Swap node values

```
#include <iostream>
using namespace std;
```

```
//node structure
struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    //Add new element at the end of the list
    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    //swap node values
    void swapNodeValues(int node1, int node2) {

      Node* temp = head;
      int N = 0;
      while(temp != NULL) {
        N++;
        temp = temp->next;
      }

      if(node1 < 1 || node1 > N || node2 < 1 || node2 > N)
        return;

      Node* pos1 = head;
      Node* pos2 = head;
      for(int i = 1; i < node1; i++) {
```

```
        pos1 = pos1->next;
      }
      for(int i = 1; i < node2; i++) {
        pos2 = pos2->next;
      }

      int val = pos1->data;
      pos1->data = pos2->data;
      pos2->data = val;
    }

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
        }
        cout<<"\n";
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

// test the code
int main() {
  LinkedList MyList;

  //Add five elements in the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.push_back(40);
  MyList.push_back(50);

  //Display the content of the list.
  MyList.PrintList();

  //swap values of node=1 and node=4
  MyList.swapNodeValues(1, 4);

  //Display the content of the list.
  MyList.PrintList();

  return 0;
```

```
}
```

The above code will give the following output:

```
The list contains: 10 20 30 40 50
The list contains: 40 20 30 10 50
```