# Circular Singly Linked List - Traversal

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

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

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

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

```
            break;
        }
        cout<<endl;
      } 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();

  return 0;
}
```

The above code will give the following output:

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

# Circular Singly Linked List - Insert a new node at the start

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

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

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;
      if(head == NULL) {
        head = newNode;
        newNode->next = head;
      } else {
        Node* temp = head;
        while(temp->next != head)
          temp = temp->next;
        temp->next = 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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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
```

# Circular Singly Linked List - Insert a new node at the end

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

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

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

```
    }

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

# Circular Singly Linked List - Insert a new node at the given position

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

//node structure
```

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

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

    //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;
      Node* temp = head;
      int NoOfElements = 0;

      if(temp != NULL) {
        NoOfElements++;
        temp = temp->next;
      }
      while(temp != head) {
        NoOfElements++;
        temp = temp->next;
      }

      if(position < 1 || position > (NoOfElements+1)) {
```

```
        cout<<"\nInavalid position.";
      } else if (position == 1) {

        if(head == NULL) {
          head = newNode;
          head->next = head;
        } else {
          while(temp->next != head) {
            temp = temp->next;
          }
          newNode->next = head;
          head = newNode;
          temp->next = head;
        }
      } else {
        temp = head;
        for(int i = 1; i < position-1; i++)
          temp = temp->next;
        newNode->next = temp->next;
        temp->next = newNode;
      }
    }

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

  //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
```

# Circular Singly Linked List - Delete the first node

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

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

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

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

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

# Circular Singly Linked List - Delete the last node

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

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

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

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

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

# Circular Singly Linked List - Delete a node at the given position

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

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

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

  //Delete an element at the given position
  void pop_at(int position) {
    Node* nodeToDelete = head;
    Node* temp = head;
    int NoOfElements = 0;

    if(temp != NULL) {
      NoOfElements++;
      temp = temp->next;
    }
    while(temp != head) {
      NoOfElements++;
      temp = temp->next;
    }

    if(position < 1 || position > NoOfElements) {
      cout<<"\nInavalid position.";
    } else if (position == 1) {
      if(head->next == head) {
        head = NULL;
      } else {
        while(temp->next != head)
          temp = temp->next;
        head = head->next;
        temp->next = head;
        free(nodeToDelete);
      }
    } else {
```

```
        temp = head;
        for(int i = 1; i < position-1; i++)
          temp = temp->next;
        nodeToDelete = temp->next;
        temp->next = temp->next->next;
        free(nodeToDelete);
      }
    }

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

# Circular Singly Linked List - Delete all nodes

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

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

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

```
    //delete all nodes of the list
    void deleteAllNodes() {
      if(head != NULL) {
        Node *temp, *current;
        current = head->next;
        while(current != head) {
          temp = current->next;
          free(current);
          current = temp;
        }
        free(head);
        head = NULL;
      }
      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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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.
```

# Circular Singly Linked List - Count nodes

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

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

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

```
        temp->next = newNode;
        newNode->next = head;
      }
    }

    //count nodes in the list
    int countNodes() {
      Node* temp = head;
      int i = 0;
      if(temp != NULL) {
        i++;
        temp = temp->next;
      }
      while(temp != head) {
        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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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
```

# Circular Singly Linked List - Delete even nodes

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

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

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

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

    //delete even nodes of the list
    void deleteEvenNodes() {
      if(head != NULL && head->next != head) {
        Node* oddNode = head;
        Node* evenNode = head->next;
        Node* temp = new Node();
        while(true) {
          temp = oddNode;
          oddNode->next = evenNode->next;
          free(evenNode);
          oddNode = oddNode->next;
          evenNode = oddNode->next;
          if(oddNode == head || evenNode == head)
            break;
        }
        if(oddNode == head)
          temp->next = head;
        else
          oddNode->next = head;
      }
    }

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

# Circular Singly Linked List - Delete odd nodes

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

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

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

  //delete odd nodes of the list
  void deleteOddNodes() {
    if(head != NULL && head->next == head) {
      free(head);
      head = NULL;
    } else if(head != NULL) {

      Node* temp = head;
      while(temp->next != head) {
        temp = temp->next;
      }
      temp->next = head->next;
      free(head);
      head = temp->next;

      if(head != NULL && head->next != head) {

        Node* evenNode = head;
        Node* oddNode = head->next;
        while(true) {
          temp = evenNode;
          evenNode->next = oddNode->next;
          free(oddNode);
          evenNode = evenNode->next;
```

```
            oddNode = evenNode->next;
            if(evenNode == head || oddNode == head)
              break;
          }

          if(evenNode == head)
            temp->next = head;
          else
            evenNode->next = head;
        }
      }
    }

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

# Circular Singly Linked List - Search an element

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

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

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

```
      temp = temp->next;
    temp->next = newNode;
    newNode->next = head;
  }
}

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

  if(temp != NULL) {
    while(true) {
      i++;
      if(temp->data == searchValue) {
        found++;
        break;
      }
      temp = temp->next;
      if(temp == head) {break;}
    }
    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(true) {
      cout<<temp->data<<" ";
      temp = temp->next;
      if(temp == head)
        break;
    }
    cout<<endl;
  } 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.
```

# Circular Singly Linked List - Delete first node by key

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

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

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

  //Delete first node by key
  void pop_first(int key) {
    if(head != NULL) {
      Node* temp = head;
      Node* nodeToDelete = head;

      if(temp->data == key) {
        if(temp->next == head) {
          head = NULL;
        } else {

          while(temp->next != head) {
            temp = temp->next;
          }
          head = head->next;
          temp->next = head;
          free(nodeToDelete);
        }
      } else {
        while(temp->next != head) {
          if(temp->next->data == key) {
            nodeToDelete = temp->next;
            temp->next = temp->next->next;
            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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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(10);
  MyList.push_back(20);
  MyList.PrintList();

  //Delete 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
```

# Circular Singly Linked List - Delete last node by key

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

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

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

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

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

    //display the content of the list
    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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(20);
  MyList.push_back(40);
  MyList.PrintList();

  //Delete 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
```

# Circular Singly Linked List - Delete all nodes by key

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

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

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

  //Delete all nodes by key
  void pop_all(int key) {
    Node* nodeToDelete;
    Node* temp;

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

    temp = head;
    if(temp != NULL) {
      while(temp->next != head) {
        if(temp->next->data == key) {
          nodeToDelete = temp->next;
          temp->next = temp->next->next;
          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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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(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
```

# Circular Singly Linked List - Reverse the List

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

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

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

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

        prevNode->next = prevNode;
```

```
        while(curNode != head) {
          tempNode = curNode->next;
          curNode->next = prevNode;
          head->next = 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(true) {
          cout<<temp->data<<" ";
          temp = temp->next;
          if(temp == head)
            break;
        }
        cout<<endl;
      } 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();

  //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
```

# Circular Singly Linked List - Swap node values

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

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

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

```
      newNode->next = head;
    }
  }

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

    Node* temp = head;
    int N = 0;
    if(temp != NULL) {
      N++;
      temp = temp->next;
    }
    while(temp != head) {
      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(true) {
        cout<<temp->data<<" ";
        temp = temp->next;
        if(temp == head)
          break;
      }
      cout<<endl;
    } 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
```