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

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

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

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

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

```
  second->next = third;

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

```
/// INSERT NODE AT BEGINING
#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 = 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;
}

///INSERT NEW NODE AT 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;
    } else {
      Node* temp = head;
      while(temp->next != NULL)
        temp = temp->next;
      temp->next = 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 end of the list.
  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.PrintList();

  return 0;
}



///INSERT NEW NODE AT 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;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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;

  if(position < 1) {
    cout<<"\nposition should be >= 1.";
  } else if (position == 1) {
    newNode->next = head;
    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;
      temp->next = 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;
}

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

    //Delete first node of the list
    void pop_front() {
```

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

///DELETE 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) {
    //for first element in the list
    head = newNode;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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;
}

///DELETE NODE AT GIEVN 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) {
    //for first element in the list
    head = newNode;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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;
}




///DELETE A NODE AT 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;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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);
  } 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;
        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;
}



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

//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;
}



///COUNT NUMBER OF 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;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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;
}




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

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



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

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



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

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

```
        free(oddNode);
        evenNode = evenNode->next;
        if(evenNode != NULL)
          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;
}
```

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

    //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;
}
```

```
///REVERSE AN LL
#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;
     } else {
       Node* temp = head;
       while(temp->next != NULL)
```

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

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

   prevNode->next = NULL;

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



///SWAP VALUES OF LL
#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;
  } else {
    Node* temp = head;
    while(temp->next != NULL)
      temp = temp->next;
    temp->next = newNode;
  }
}

//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;
}
```