Answer on Question#39348 - Programming - C++

```
#include <iostream>

using namespace std;

class list
{
public:
    struct node {
        int value;
        struct node *next;
    } *head, *tail, *ptr;

    list():head(NULL),tail(NULL){}      // constructor
    ~list();                           // destructor

    struct list::node* initNode(int);

    void insertNode(int value);
    struct list::node* locateToNodeWithValue(int);
    void deleteNode( int value);
    void printAllEvenNumbers();

    void displayList( struct list::node*)const ;
    void displayNode( struct list::node*) const;
};

list::~list() {
    node *current,*temp;
    current = head;
    temp = head;
    while(current != NULL) {
        current = current->next;
        delete temp;
        temp = current;
    }
}

struct list::node* list::initNode( int v) {
    struct node *ptr = new node;

    // error? then just return
    if( ptr == NULL )
        return static_cast<struct node *>(NULL);
    // assign it
    // then return pointer to the node
    else {
        ptr->value = v ;

        return ptr;
    }
}
```

```
// adding to the end of list
void list::insertNode( int value )  {
    struct node *newNode = this->initNode( value );

    // if there is no node, put it to head
    if( head == NULL ) {
        head = newNode;
        tail = newNode;
    }

    // link in the new_node to the tail of the list
    // then mark the next field as the end of the list
    // adjust tail to point to the last node

    tail->next = newNode;
    newNode->next = NULL;
    tail = newNode;
}

struct list::node* list::locateToNodeWithValue( int value) {
    ptr=head;
    while(ptr!=NULL) {
       if(value==ptr->value)
          return ptr;
        ptr = ptr->next;
    }
    cout<<"Element with value "<< value<<" not found"<<endl;
    return NULL;
}

void list::deleteNode( int value )
{
  ptr=locateToNodeWithValue(value);
  if (ptr==NULL){
      return;
  }

  struct node *temp, *prev;
   temp = ptr;    // node to be deleted
   prev = head;   // start of the list, will cycle to node before temp

   if( temp == prev ) {                    // deleting first node?
       head = head->next;                  // moves head to next node
       if( tail == temp )                  // is it end, only one node?
          tail = tail->next;               // adjust end as well
       delete temp ;                       // free up space
   }
   else {                                  // if not the first node, then
       while( prev->next != temp ) {       // move prev to the node before
           prev = prev->next;              // the one to be deleted
       }
       prev->next = temp->next;            // link previous node to next
       if( tail == temp )                  // if this was the end node,
           tail = prev;                    // then reset the end pointer
      delete temp;                         // free up space
   }
```

```
}

void list::printAllEvenNumbers(){
    list::node* ptr=head;
    while(ptr!=NULL){
        if ((ptr->value % 2)==0)
            displayNode(ptr);
        ptr=ptr->next;
    }
}

void list::displayNode( struct list::node *ptr ) const
{
    cout << "node: "<<ptr->value << endl;
}

void list::displayList( struct list::node *ptr ) const
{
    if(!ptr) cout << "Nothing to display" << endl;
    while(ptr) {
        displayNode(ptr);
        ptr = ptr->next;
    }
}

int main()
{
    int value;

    list myList;

    list::node* ptr;

    //2. Insert a node to a list

    myList.insertNode(1);
    myList.insertNode(13);
    myList.insertNode(4);
    myList.insertNode(6);
    myList.insertNode(42);
    myList.insertNode(12);

    myList.displayList(myList.head);

    value=42;

    //3.locate to an element in a list by value
    list::node* fortyTwoPtr=myList.locateToNodeWithValue(value);

    cout<<"The next node after the node with vlaue 42 has value "<<fortyTwoPtr-
>next->value<<endl;

    //4. delete element by value
    //from middle
    myList.deleteNode(42);
```

```
    cout<<endl<<"deleted 42:"<<endl;
    myList.displayList(myList.head);

    //from start
    myList.deleteNode(1);
    cout<<endl<<"deleted 1:"<<endl;
    myList.displayList(myList.head);

    //from end
    myList.deleteNode(12);
    cout<<endl<<"deleted 12:"<<endl;
    myList.displayList(myList.head);

    //attempt to delete unexisting node
    myList.deleteNode(46);
    myList.displayList(myList.head);

    cout<<"print even numbers:"<<endl;
    myList.printAllEvenNumbers();

    return 0;
}
```