Single linked list:

Headfile:

```
#ifndef HW5_H
#define HW5_H
#pragma once
#include <iostream>
class hw5
{
public:
    class Node {
        public:
        int data;
        Node* next;
    };
    hw5();
    void addElement(int data);
    void removeElement(int data);
    void searchElement(int data);
    void printList();
    ~hw5();

private:
    Node* head;

};

#endif
```

Cpp file:

```
#include "hw5.h"
#include <iostream>
using namespace std;
hw5::hw5()
{
    head = NULL;
}
void hw5::searchElement(int data){
    Node* temp = head;
        while (temp != NULL && temp->data != data) {
            temp = temp->next;
        }
        if (temp == NULL) {
            cout << "Element " << data << " not found in the list." << endl;
        }
        else {
            cout << "Element " << data << " found in the list." << endl;
        }
}
void hw5::addElement(int data){
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
    }
    else {
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    cout << "Element " << data << " added to the list." << endl;
}
void hw5::removeElement(int data){
    Node* temp = head;
        Node* prev = NULL;
        while (temp != NULL && temp->data != data) {
            prev = temp;
            temp = temp->next;
        }
        if (temp == NULL) {
            cout << "Element " << data << " not found in the list." << endl;
            return;
        }
        if (prev == NULL) {
            head = temp->next;
        }
        else {
            prev->next = temp->next;
        }
        delete temp;
        cout << "Element " << data << " removed from the list." << endl;
}
void hw5::printList(){
    if (head == NULL) {
            cout << "List is empty." << endl;
        }
        else {
            cout << "List elements: ";
            Node* temp = head;
            while (temp != NULL) {
                cout << temp->data << " ";
                temp = temp->next;
            }
            cout << endl;
        }
}
hw5::~hw5()
{

}
```

Doubly linked list:

headFile:

```
#ifndef DOUBLY_LINKED_LIST_H
#define DOUBLY_LINKED_LIST_H

#pragma once

class doubly_linked_list
{
public:
    class Node {
    public:
        int data;
        Node* next;
        Node* prev;
    };
    doubly_linked_list();
    void addElement(int data);
    void removeElement(int data);
    void searchElement(int data);
    void printList();
    ~doubly_linked_list();

private:
    Node* head;
    Node* tail;
};

#endif
```

cppFile:

```cpp
#include "doubly_linked_list.h"
#include <iostream>
using namespace std;
doubly_linked_list::doubly_linked_list()
{
    head = NULL;
    tail = NULL;
}
void doubly_linked_list::addElement(int data) {
        Node* newNode = new Node;
        newNode->data = data;
        newNode->next = NULL;
        newNode->prev = tail;
        if (head == NULL) {
            head = newNode;
        }
        else {
            tail->next = newNode;
        }
        tail = newNode;
        cout << "Element " << data << " added to the list." << endl;
    }
void doubly_linked_list::removeElement(int data) {
        Node* temp = head;
        while (temp != NULL && temp->data != data) {
            temp = temp->next;
        }
        if (temp == NULL) {
            cout << "Element " << data << " not found in the list." << endl;
            return;
        }
        if (temp == head && temp == tail) {
            head = NULL;
            tail = NULL;
        }
        else if (temp == head) {
            head = temp->next;
            head->prev = NULL;
        }
        else if (temp == tail) {
            tail = temp->prev;
            tail->next = NULL;
        }
        else {
            temp->prev->next = temp->next;
            temp->next->prev = temp->prev;
        }
        delete temp;
        cout << "Element " << data << " removed from the list." << endl;
    }
void doubly_linked_list::searchElement(int data) {
        Node* temp = head;
        while (temp != NULL && temp->data != data) {
            temp = temp->next;
        }
        if (temp == NULL) {
            cout << "Element " << data << " not found in the list." << endl;
        }
        else {
            cout << "Element " << data << " found in the list." << endl;
        }
    }

    // function to print the elements of the list
void doubly_linked_list::printList() {
    if (head == NULL) {
        cout << "List is empty." << endl;
    }
    else {
        cout << "List elements: ";
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
    }
doubly_linked_list::~doubly_linked_list()
{

}
```

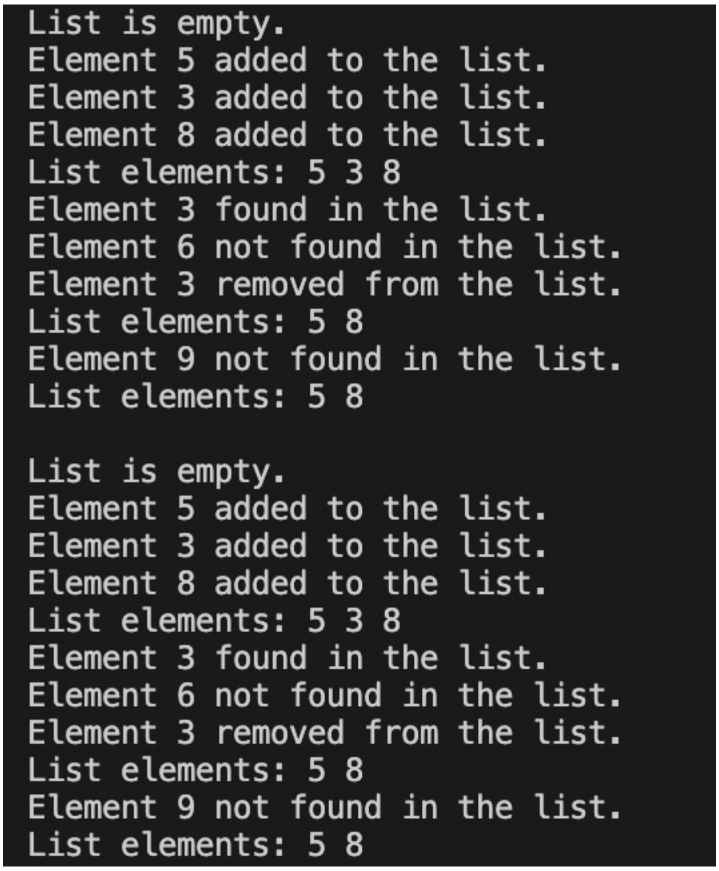
mainProgram:

```
#include <iostream>
#include "hw5.cpp"
#include "doubly_linked_list.cpp"
```

```
using namespace std;
int main(){
    hw5 myList;
    myList.printList();
    myList.addElement(5);
    myList.addElement(3);
    myList.addElement(8);
    myList.printList();
    myList.searchElement(3);
    myList.searchElement(6);
    myList.removeElement(3);
    myList.printList();
    myList.removeElement(9);
    myList.printList();
    cout << endl;

    doubly_linked_list myList2;
    myList2.printList();
    myList2.addElement(5);
    myList2.addElement(3);
    myList2.addElement(8);
    myList2.printList();
    myList2.searchElement(3);
    myList2.searchElement(6);
    myList2.removeElement(3);
    myList2.printList();
    myList2.removeElement(9);
    myList2.printList();
    return 0;
}
```

Result:



```
List is empty.
Element 5 added to the list.
Element 3 added to the list.
Element 8 added to the list.
List elements: 5 3 8
Element 3 found in the list.
Element 6 not found in the list.
Element 3 removed from the list.
List elements: 5 8
Element 9 not found in the list.
List elements: 5 8

List is empty.
Element 5 added to the list.
Element 3 added to the list.
Element 8 added to the list.
List elements: 5 3 8
Element 3 found in the list.
Element 6 not found in the list.
Element 3 removed from the list.
List elements: 5 8
Element 9 not found in the list.
List elements: 5 8
```