```cpp
#include <iostream>


using namespace std;


struct Node {
    int data;
    Node* prev = nullptr;
    Node* next = nullptr;
};


Node* head = nullptr;


Node* newNode(int data) {
    Node* node = new Node;
    node->data = data;
    return node;
}


void insertIntoEmptyList(int data) {
    if( head != nullptr ) {
        return;
    }
    Node* node = newNode(data);
    head = node;
    head->next = head;
    head->prev = head;
}


void insertAtHead(int data) {
    if(head == nullptr) {
        insertIntoEmptyList(data);
        return;
    }
    Node* node = newNode(data);
    node->next = head;
    node->prev = head->prev;
    head->prev->next = node;
    head->prev = node;
    head = node;
}


void insertAtTail(int data) {
    if(head == nullptr) {
        insertIntoEmptyList(data);
        return;
    }
    Node* node = newNode(data);
    Node* tail = head->prev;
    node->prev = tail;
    node->next = head;
    tail->next = node;
    head->prev = node;
}



void insertAtMiddleAfter(int after, int data) {
    if (head == nullptr) {
        return;
    }
    if(head != nullptr && head->prev != nullptr && head->prev->data ==  after) {
        insertAtTail(data);
        return;
    }
    Node* prev = head;
    while(prev != head->prev) {
        if(prev->data == after) {
            Node* node = newNode(data);
            node->next = prev->next;
            node->prev = prev;
            prev->next = node;
            node->next->prev = node;
            break;
        }
        prev = prev->next;
    }
}



void deleteFromHead() {
    if(head == nullptr) {
        return;
    }
    Node* node = head;
    head = node->next;
    if (head != node) {
        //list will not be empty after deleting head element
        head->prev = node->prev;
        node->prev->next = head;
    }
    else {
        //list will be empty after deleting the only element.
        head = nullptr;
    }
    delete node;
}
```

```cpp
void deleteFromTail() {
    if(head == nullptr) {
        return;
    }
    Node* tail = head->prev;
    head->prev = tail->prev;
    if(head->prev != tail) {
        //list will not be empty after deleting the tail element.
        head->prev->next = head;
    }
    else {
        //list will be empty after deleting the only element.
        head = nullptr;
    }
    delete tail;
}



void deleteFromMiddle(int data) {
    if(head == nullptr ) {
        return;
    }
    if (head->data == data) {
        deleteFromHead();
        return;
    }
    if(head->prev->data == data) {
        deleteFromTail();
        return;
    }
    Node* prev = head;
    while(prev != head->prev) {
        if(prev->next != nullptr && prev->next->data == data) {
            Node* node = prev->next;
            prev->next = node->next;
            node->next->prev = prev;
            delete node;
            break;
        }
        prev = prev->next;
    }
}



void showForward() {
    if(head == nullptr) {
        cout<<endl;
        return;
    }
    Node* node = head;
    while(node != head->prev) {
        cout<<" "<<node->data;
        node = node->next;
    }
    cout<<" "<<node->data<<endl;
}



void showBackward() {
    if(head == nullptr) {
        cout<<endl;
        return;
    }

    Node* node = head->prev;
    while(node != head) {
        cout<<" "<<node->data;
        node = node->prev;
    }
    cout<<" "<<node->data<<endl;
}



int main() {

    insertAtHead(10);
    insertAtHead(20);
    insertAtHead(30);
    insertAtHead(40);
    insertAtHead(50);
    showForward();
    showBackward();

    insertAtMiddleAfter(30, 60);
    showForward();
    showBackward();

    insertAtMiddleAfter(10, 70);
    showForward();
    showBackward();

    insertAtTail(80);
    showForward();
    showBackward();


    deleteFromMiddle(60);
    showForward();
    showBackward();

    deleteFromMiddle(70);
    showForward();
```

```
    showBackward();

    deleteFromHead();
    showForward();
    showBackward();

    while(head != nullptr) {
        deleteFromTail();
        showForward();
        showBackward();
    }
    return 0;
}
```