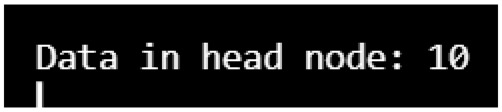
#1

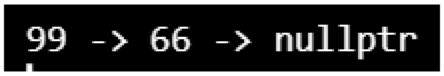
```
#include <iostream>
using namespace std;
struct Node {
int data;
    Node* next;
};
int main() {
    Node* head = new Node();
    head->data = 10;
    head->next = nullptr;
    cout << "Data in head node: " << head->data << endl;
    delete head;
    return 0;
}
```



```
Data in head node: 10
```

#2

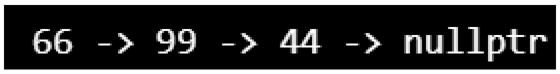
```
#include <iostream>
using namespace std;
struct Node {
int data;
Node* next;
};
void insertAtBeginning(Node*& head, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "nullptr" << endl;
    }
int main() {
    Node* head = nullptr;
    insertAtBeginning(head, 66);
    insertAtBeginning(head, 99);
    printList(head);

    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    return 0;
}
```



```
99 -> 66 -> nullptr
```

#3

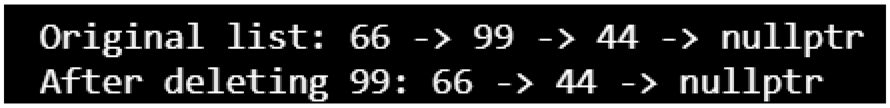
```
#include <iostream>
using namespace std;
struct Node {
int data;
Node* next;
};
void insertAtBeginning(Node*& head, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}
void insertAtEnd(Node*& head, Node*& tail, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = nullptr; // New node will be the last, so next is nullptr
    if (tail == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "nullptr" << endl;
    }
int main() {
    Node* head = nullptr;
    Node* tail = nullptr;
    insertAtEnd(head, tail, 66);
    insertAtEnd(head, tail, 99);
    insertAtEnd(head, tail, 44);
    printList(head);

    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    return 0;
}
```



```
66 -> 99 -> 44 -> nullptr
```

#4

```
#include <iostream>
using namespace std;
struct Node {
int data;
Node* next;
};
void insertAtBeginning(Node*& head, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = head;
    head = newNode;
}
void insertAtEnd(Node*& head, Node*& tail, int newData) {
    Node* newNode = new Node();
    newNode->data = newData;
    newNode->next = nullptr;
    if (tail == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}
void deleteNode(Node*& head, Node*& tail, int key) {
    if (head == nullptr) return;
    if (head->data == key) {
        Node* temp = head;
        head = head->next;
        if (head == nullptr) {
            tail = nullptr;
        }
        delete temp;
        return;
    }
    Node* temp = head;
    while (temp->next != nullptr && temp->next->data != key) {
        temp = temp->next;
    }
    if (temp->next == nullptr) return;
    Node* toDelete = temp->next;
    temp->next = temp->next->next;

    if (toDelete == tail) {
        tail = temp;
    }
    delete toDelete;
}
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "nullptr" << endl;
    }
int main() {
    Node* head = nullptr;
    Node* tail = nullptr;
    insertAtEnd(head, tail, 66);
    insertAtEnd(head, tail, 99);
    insertAtEnd(head, tail, 44);
    cout << "Original list: ";
    printList(head);
    deleteNode(head, tail, 99);
    cout << "After deleting 99: ";
    printList(head);

    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }
    return 0;
}
```

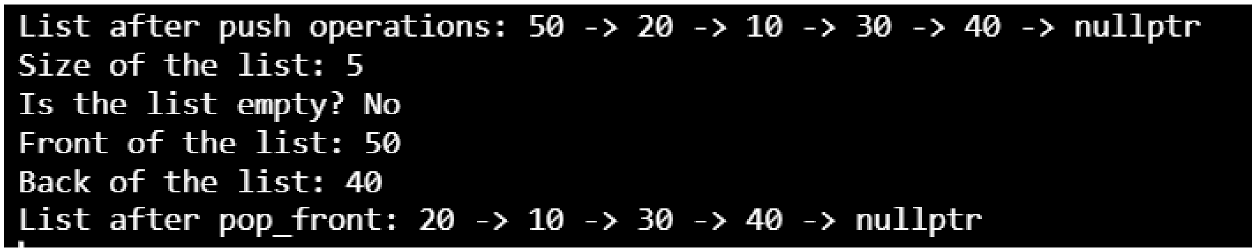


```
Original list: 66 -> 99 -> 44 -> nullptr
After deleting 99: 66 -> 44 -> nullptr
```

#5

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
class SinglyLinkedList {
    private:
        Node* head;
        Node* tail;
        int listSize;
    public:
        SinglyLinkedList() : head(nullptr), tail(nullptr), listSize(0) {}

        bool empty() {
            return listSize == 0;
        }

        int size() {
            return listSize;
        }

        int front() {
            if (head != nullptr) {
                return head->data;
            }
            throw runtime_error("List is empty!");
        }

        int back() {
            if (tail != nullptr) {
                return tail->data;
            }
            throw runtime_error("List is empty!");
        }

        void push_front(int newData) {
            Node* newNode = new Node();
            newNode->data = newData;
            newNode->next = head;
            head = newNode;
            if (tail == nullptr) {
                tail = newNode;
            }
            listSize++;
        }

        void push_back(int newData) {
            Node* newNode = new Node();
            newNode->data = newData;
            newNode->next = nullptr;
            if (tail == nullptr) {
                head = newNode;
                tail = newNode;
            } else {
                tail->next = newNode;
                tail = newNode;
            }
            listSize++;
        }

        void pop_front() {
            if (head == nullptr) {
                throw runtime_error("Cannot pop from an empty list!");
            }
            Node* temp = head;
            head = head->next;
            delete temp;
            listSize--;
            if (head == nullptr) {
```

```
                tail = nullptr;
            }
        }

        void printList() {
            if (empty()) {
                cout << "List is empty." << endl;
                return;
            }
            Node* temp = head;
            while (temp != nullptr) {
                cout << temp->data << " -> ";
                temp = temp->next;
            }
            cout << "nullptr" << endl;
        }

        ~SinglyLinkedList() {
            while (head != nullptr) {
            Node* temp = head;
            head = head->next;
            delete temp;
            }
        }
    };
int main() {
    SinglyLinkedList list;

    list.push_back(10); // List: 10
    list.push_front(20); // List: 20 -> 10
    list.push_back(30); // List: 20 -> 10 -> 30
    list.push_back(40); // List: 20 -> 10 -> 30 -> 40
    list.push_front(50); // List: 50 -> 20 -> 10 -> 30 -> 40
    cout << "List after push operations: ";
    list.printList();

    cout << "Size of the list: " << list.size() << endl;
    cout << "Is the list empty? " << (list.empty() ? "Yes" : "No") << endl;

    cout << "Front of the list: " << list.front() << endl;
    cout << "Back of the list: " << list.back() << endl;

    list.pop_front();
    cout << "List after pop_front: ";
    list.printList();
    return 0;
}
```



```
List after push operations: 50 -> 20 -> 10 -> 30 -> 40 -> nullptr
Size of the list: 5
Is the list empty? No
Front of the list: 50
Back of the list: 40
List after pop_front: 20 -> 10 -> 30 -> 40 -> nullptr
```