#1

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

struct node {
   int data;
   node* next;
};
class SLinkedList {
   private:
      node* head;
      node* tail;
   public:
      SLinkedList() {
         head = nullptr;
         tail = nullptr;
         cout << "head and tail nodes are initiated with nullpoint" << endl;
      }


void ListAppend(int elem) {
   node *newNode = new node;
   newNode->data = elem;
   newNode->next = nullptr;
   if (head == nullptr ) {
      head = newNode;
      tail = newNode;
   }
   else {
      tail->next = newNode;
      tail = newNode;
   }
}
void ListPrepend(int elem) {
   node* newNode = new node;
   newNode->data = elem;
   newNode->next = nullptr;
   if (head == nullptr) {
      head = newNode;
      tail = newNode;
   }
   else {
      newNode->next = head;
```

```
        head = newNode;
    }
}

void ListDisplay() {
    node *tmp;
    tmp = head;
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

void InsertAfter(node* curNode, int elem) {
    node* newNode = new node;
    newNode->data = elem;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    }

    else if (curNode == tail) {
        tail->next = newNode;
        tail = newNode;
    }

    else {
        newNode->next = curNode->next;
        curNode->next = newNode;
    }
}

node* GetNode(int value) {
    node* tmp = head;
    while (tmp != nullptr) {
        if (tmp->data == value) {
            return tmp;
        }
        tmp = tmp->next;
    }
    return nullptr;
```

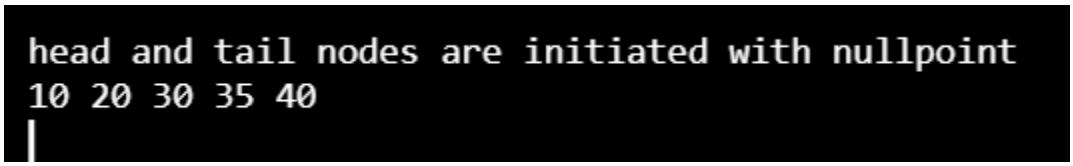
```
}

};

int main() {
    SLinkedList numList1;
    numList1.ListAppend(30);
    numList1.ListAppend(40);
    numList1.ListPrepend(20);
    numList1.ListPrepend(10);

    node* curNode = numList1.GetNode(30);
    if (curNode != nullptr) {
        numList1.InsertAfter(curNode, 35);
    }


    numList1.ListDisplay();
    return 0;
}
```



#2

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

struct node {
    int data;
    node* next;
};
class SLinkedList {
    private:
        node* head;
        node* tail;
    public:
        SLinkedList() {
            head = nullptr;
            tail = nullptr;
            cout << "head and tail nodes are initiated with nullpoint" << endl;
```

```
        }


void ListAppend(int elem) {
    node *newNode = new node;
    newNode->data = elem;
    newNode->next = nullptr;
    if (head == nullptr ) {
        head = newNode;
        tail = newNode;
    }
    else {
        tail->next = newNode;
        tail = newNode;
    }
}
void ListPrepend(int elem) {
    node* newNode = new node;
    newNode->data = elem;
    newNode->next = nullptr;
    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    }
    else {
        newNode->next = head;
        head = newNode;
    }
}

void ListDisplay() {
    node *tmp;
    tmp = head;
    while (tmp != nullptr) {
        cout << tmp->data << " ";
        tmp = tmp->next;
    }
    cout << endl;
}

void InsertAfter(node* curNode, int elem) {
    node* newNode = new node;
    newNode->data = elem;
    newNode->next = nullptr;
```

```
    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    }

    else if (curNode == tail) {
        tail->next = newNode;
        tail = newNode;
    }

    else {
        newNode->next = curNode->next;
        curNode->next = newNode;
    }
}

node* GetNode(int value) {
    node* tmp = head;
    while (tmp != nullptr) {
        if (tmp->data == value) {
            return tmp;
        }
        tmp = tmp->next;
    }
    return nullptr;
}

void RemoveAfter(node* curNode) {
    if (head == nullptr) {
        cout << "List is empty. Nothing to remove." << endl;
        return;
    }

    if (curNode == nullptr && head != nullptr) {
        node* sucNode = head->next;
        delete head;
        head = sucNode;
        if (head == nullptr) {
            tail = nullptr;
        }
        return;
    }
    if (curNode->next != nullptr) {
```

```
        node* sucNode = curNode->next;
        curNode->next = sucNode->next;

        if (sucNode == tail) {
            tail = curNode;
        }
        delete sucNode;
    }
}

};

int main() {
    SLinkedList numList1;
    numList1.ListAppend(30);
    numList1.ListAppend(40);
    numList1.ListPrepend(20);
    numList1.ListPrepend(10);

    node* curNode = numList1.GetNode(30);
    if (curNode != nullptr) {
        numList1.RemoveAfter(curNode);
    }


    numList1.ListDisplay();
    return 0;
}
```

```
head and tail nodes are initiated with nullpoint
10 20 30
```