AQ1.cpp

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

struct Node
{
   int data;
   Node *next;
};

Node *head=nullptr;

void insertAtBeginning(int val)
{
   Node *newNode=new Node();
   newNode->data=val;
   newNode->next=head;
   head=newNode;
   cout<<val<<" inserted at the beginning.\n";
}
```

```
void insertAtEnd(int val)
{
    Node *newNode=new Node();
    newNode->data=val;
    newNode->next=nullptr;
    if (head==nullptr)
    {
        head=newNode;
    }
    else
    {
        Node *temp=head;
        while (temp->next != nullptr)
            temp=temp->next;
        temp->next=newNode;
    }
    cout<<val<<" inserted at the end.\n";
}

void insertBeforeAfter(int val, int target, bool after)
```

```
{
    Node *newNode=new Node();
    newNode->data=val;

    if (head==nullptr)
    {
        cout<<"List is empty. Cannot insert.\n";
        delete newNode;
        return;
    }

    Node *temp=head;
    Node *prev=nullptr;
    while (temp != nullptr && temp->data != target)
    {
        prev=temp;
        temp=temp->next;
    }

    if (temp==nullptr)
    {
```

```
    cout<<"Target node not found.\n";
    delete newNode;
    return;
}

if (after)
{
    newNode->next=temp->next;
    temp->next=newNode;
    cout<<val<<" inserted after "<<target<<".\n";
}
else
{
    if (temp==head)
    {
        newNode->next=head;
        head=newNode;
    }
    else
    {
        newNode->next=temp;
```

```
            prev->next=newNode;
        }
        cout<<val<<" inserted before "<<target<<".\n";
    }
}

void deleteFromBeginning()
{
    if (head==nullptr)
    {
        cout<<"List is empty.\n";
        return;
    }
    Node *temp=head;
    head=head->next;
    cout<<temp->data<<" deleted from beginning.\n";
    delete temp;
}

void deleteFromEnd()
{
```

```
if (head==nullptr)
{
    cout<<"List is empty.\n";
    return;
}
if (head->next==nullptr)
{
    cout<<head->data<<" deleted from end.\n";
    delete head;
    head=nullptr;
    return;
}
Node *temp=head;
Node *prev=nullptr;
while (temp->next != nullptr)
{
    prev=temp;
    temp=temp->next;
}
prev->next=nullptr;
cout<<temp->data<<" deleted from end.\n";
```

```
    delete temp;
}

void deleteSpecificNode(int val)
{
    if (head==nullptr)
    {
        cout<<"List is empty.\n";
        return;
    }
    Node *temp=head;
    Node *prev=nullptr;

    while (temp != nullptr && temp->data != val)
    {
        prev=temp;
        temp=temp->next;
    }

    if (temp==nullptr)
    {
```

```
        cout<<"Node "<<val<<" not found.\n";
        return;
    }

    if (temp==head)
    {
        head=head->next;
    }
    else
    {
        prev->next=temp->next;
    }
    cout<<"Node "<<val<<" deleted.\n";
    delete temp;
}

void searchNode(int val)
{
    Node *temp=head;
    int pos=1;
    while (temp != nullptr)
```

```
    {
        if (temp->data==val)
        {
            cout<<"Node "<<val<<" found at position
"<<pos<<".\n";
            return;
        }
        temp=temp->next;
        pos++;
    }
    cout<<"Node "<<val<<" not found.\n";
}

void displayList()
{
    if (head==nullptr)
    {
        cout<<"List is empty.\n";
        return;
    }
    Node *temp=head;
```

```
    cout<<"Linked List: ";
    while (temp != nullptr)
    {
        cout<<temp->data<<" ";
        temp=temp->next;
    }
    cout<<endl;
}

int main()
{
    int choice, val, target;
    bool after;

    do
    {
        cout<<"Singly Linked List Menu \n";
        cout<<"1. Insert at Beginning\n";
        cout<<"2. Insert at End\n";
        cout<<"3. Insert Before/After a Node\n";
        cout<<"4. Delete from Beginning\n";
```

```
cout<<"5. Delete from End\n";
cout<<"6. Delete a Specific Node\n";
cout<<"7. Search for a Node\n";
cout<<"8. Display List\n";
cout<<"9. Exit\n";
cout<<"Enter your choice: ";
cin >> choice;

switch (choice)
{
case 1:
   cout<<"Enter value to insert: ";
   cin >> val;
   insertAtBeginning(val);
   break;
case 2:
   cout<<"Enter value to insert: ";
   cin >> val;
   insertAtEnd(val);
   break;
case 3:
```

```
    cout<<"Enter value to insert: ";
    cin >> val;
    cout<<"Enter target node value: ";
    cin >> target;
    cout<<"Insert after target? (1 for yes, 0 for before): ";
    cin >> after;
    insertBeforeAfter(val, target, after);
    break;
  case 4:
    deleteFromBeginning();
    break;
  case 5:
    deleteFromEnd();
    break;
  case 6:
    cout<<"Enter value to delete: ";
    cin >> val;
    deleteSpecificNode(val);
    break;
  case 7:
    cout<<"Enter value to search: ";
```

```
            cin >> val;
            searchNode(val);
            break;
        case 8:
            displayList();
            break;
        case 9:
            cout<<"Exited";
            break;
        default:
            cout<<"Invalid choice. Try again.\n";
        }
    } while (choice != 9);

    return 0;
}
```

```
Singly Linked List Menu
1. Insert at Beginning
2. Insert at End
3. Insert Before/After a Node
4. Delete from Beginning
5. Delete from End
6. Delete a Specific Node
7. Search for a Node
8. Display List
9. Exit
Enter your choice: 1
Enter value to insert: 22
22 inserted at the beginning.
Singly Linked List Menu
1. Insert at Beginning
2. Insert at End
3. Insert Before/After a Node
4. Delete from Beginning
5. Delete from End
6. Delete a Specific Node
7. Search for a Node
8. Display List
9. Exit
Enter your choice: 1
Enter value to insert: 55
55 inserted at the beginning.
Singly Linked List Menu
1. Insert at Beginning
2. Insert at End
3. Insert Before/After a Node
4. Delete from Beginning
5. Delete from End
6. Delete a Specific Node
7. Search for a Node
8. Display List
9. Exit
Enter your choice: 1
Enter value to insert: 77
77 inserted at the beginning.
Singly Linked List Menu
1. Insert at Beginning
2. Insert at End
3. Insert Before/After a Node
4. Delete from Beginning
5. Delete from End
6. Delete a Specific Node
7. Search for a Node
8. Display List
9. Exit
Enter your choice: 8
Linked List: 77 55 22
Singly Linked List Menu
1. Insert at Beginning
2. Insert at End
3. Insert Before/After a Node
4. Delete from Beginning
5. Delete from End
6. Delete a Specific Node
7. Search for a Node
8. Display List
9. Exit
Enter your choice: 9
Exited
PS F:\Work\SEM3\DSA\5>
```

AQ2.cpp

```
return 0;
#include <iostream>
using namespace std;

struct Node
{
    int data;
    Node *next;
};

Node *head=nullptr;

void insertAtEnd(int val)
{
    Node *newNode=new Node();
    newNode->data=val;
    newNode->next=nullptr;
    if (head==nullptr)
    {
```

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

int countKey(int key)
{
    int count=0;
    Node *temp=head;
    while (temp != nullptr)
    {
        if (temp->data==key)
            count++;
        temp=temp->next;
    }
    return count;
}
```

```
void deleteAllOccurrences(int key)
{
    while (head != nullptr && head->data==key)
    {
        Node *temp=head;
        head=head->next;
        delete temp;
    }
    Node *curr=head;
    Node *prev=nullptr;
    while (curr != nullptr)
    {
        if (curr->data==key)
        {
            prev->next=curr->next;
            delete curr;
            curr=prev->next;
        }
        else
        {
```

```
            prev=curr;
            curr=curr->next;
        }
    }
}

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

```
    cout<<endl;
}

int main()
{
    int n, val, key;
    cout<<"Enter number of elements in linked list: ";
    cin >> n;
    cout<<"Enter elements:\n";
    for (int i=0; i < n; i++)
    {
        cin >> val;
        insertAtEnd(val);
    }

    cout<<"Enter key to count and delete: ";
    cin >> key;

    int count=countKey(key);
    cout<<"Count: "<<count<<endl;
```

```
    deleteAllOccurrences(key);

    cout<<"Updated Linked List: ";
    displayList();

    return 0;
}
```

```
Enter number of elements in linked list: 7
Enter elements:
1
2
1
2
1
3
1
Enter key to count and delete: 1
Count: 4
Updated Linked List: 2->2->3
PS F:\Work\SEM3\DSA\5>
```

AQ3.cpp

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

class Node {
public:
```

```
    int data;
    Node* next;

    Node(int x) {
        this->data = x;
        this->next = nullptr;
    }
};

int getLength(Node* head) {
    int length = 0;
    while (head) {
        length++;
        head = head->next;
    }
    return length;
}

int getMiddle(Node* head) {
    int length = getLength(head);
    int midIndex = length / 2;
```

```
    while (midIndex--) {
        head = head->next;
    }
    return head->data;
}

void insertAtEnd(Node*& head, int val) {
    Node* newNode = new Node(val);
    if (head == nullptr) {
        head = newNode;
        return;
    }
    Node* temp = head;
    while (temp->next != nullptr)
        temp = temp->next;
    temp->next = newNode;
}

void displayList(Node* head) {
    while (head != nullptr) {
        cout << head->data;
```

```
        if (head->next != nullptr) cout << "->";
        head = head->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int n, val;

    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter elements:\n";
    for (int i = 0; i < n; i++) {
        cin >> val;
        insertAtEnd(head, val);
    }

    cout << "Linked List: ";
    displayList(head);
```

```
    cout << "Middle element: " << getMiddle(head) << endl;


    return 0;
}
```

```
Enter number of elements: 5
Enter elements:
1
2
3
4
5
Linked List: 1->2->3->4->5
Middle element: 3
PS F:\Work\SEM3\DSA\5>
```

AQ4.cpp

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


class Node
{
public:
    int data;
    Node *next;


    Node(int x)
```

```
    {
        data=x;
        next=nullptr;
    }
};

void insertAtEnd(Node *&head, int val)
{
    Node *newNode=new Node(val);
    if (head==nullptr)
    {
        head=newNode;
        return;
    }
    Node *temp=head;
    while (temp->next != nullptr)
        temp=temp->next;
    temp->next=newNode;
}

void displayList(Node *head)
```

```
{
    while (head != nullptr)
    {
        cout<<head->data;
        if (head->next != nullptr)
            cout<<"->";
        head=head->next;
    }
    cout<<"->NULL"<<endl;
}

void reverseList(Node *&head)
{
    Node *prev=nullptr;
    Node *curr=head;
    Node *nextNode=nullptr;

    while (curr != nullptr)
    {
        nextNode=curr->next;
        curr->next=prev;
```

```
        prev=curr;
        curr=nextNode;
    }
    head=prev;
}

int main()
{
    Node *head=nullptr;
    int n, val;

    cout<<"Enter number of elements: ";
    cin >> n;
    cout<<"Enter elements:\n";
    for (int i=0; i < n; i++)
    {
        cin >> val;
        insertAtEnd(head, val);
    }

    cout<<"Original Linked List: ";
```

```
    displayList(head);

    reverseList(head);

    cout<<"Reversed Linked List: ";
    displayList(head);

    return 0;
}
```

```
Enter number of elements: 4
Enter elements:
11
22
33
44
Original Linked List: 11->22->33->44->NULL
Reversed Linked List: 44->33->22->11->NULL
PS F:\Work\SEM3\DSA\5>
```