**Sinlgy - Linked List Insertion:**

1. **At Beginning:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node \*next;

};

void insertBegin (Node \*\*head, int value) {

Node \*newNode = new Node;

newNode->data = value;

newNode->next = \*head;

\*head = newNode;

}

void printList(Node \*head) {

while (head != NULL) {

cout << head->data << " " << endl;

head = head->next;

} }

int main(){

Node\* head = NULL;

insertBegin(&head, 5);

insertBegin(&head, 10);

insertBegin(&head, 15);

printList(head);

return 0;

}

1. **At End:**

#include <iostream>

using namespace std;

// Define a struct for a node

struct Node {

int data;

Node\* next;

};

// Function to insert a new node at the end of the linked list

void insertAtEnd(Node\*& head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = NULL;

if (head == NULL) {

// If the list is empty, make the new node the head

head = newNode;

} else {

// Traverse to the end of the list

Node\* current = head;

while (current->next != NULL) {

current = current->next;

}

// Insert the new node at the end

current->next = newNode;

} }

// Function to display the linked list

void displayList(Node\* head) {

Node\* current = head;

while (current != NULL) {

cout << current->data << " ";

current = current->next;

} }

int main() {

Node\* head = NULL; // Initialize an empty linked list

// Insert nodes at the end of the list

insertAtEnd(head, 1);

insertAtEnd(head, 2);

insertAtEnd(head, 3);

insertAtEnd(head, 4);

// Display the linked list

displayList(head);

return 0;

}

1. **At any Position**

#include <iostream>

using namespace std;

// Define a struct for a node

struct Node {

int data;

Node\* next;

};

// Function to insert a new node at a specified position in the linked list

void insertAtPosition(Node\*& head, int value, int position) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (position == 0) {

// Insert at the beginning

newNode->next = head;

head = newNode;

} else {

// Traverse to the position

Node\* current = head;

int currentPosition = 0;

while (currentPosition < position - 1 && current->next != nullptr) {

current = current->next;

currentPosition++;

}

// Insert the new node at the specified position

newNode->next = current->next;

current->next = newNode;

}

}

// Function to display the linked list

void displayList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

} }

int main() {

Node\* head = nullptr; // Initialize an empty linked list

// Insert nodes at various positions in the list

insertAtPosition(head, 1, 0); // Insert at the beginning

insertAtPosition(head, 2, 1); // Insert at position 1

insertAtPosition(head, 4, 1); // Insert at position 1

insertAtPosition(head, 3, 2); // Insert at position 2

// Display the linked list

cout << "Linked List: ";

displayList(head);

return 0;

}

**Sinlgy - Linked List Deletion**

1. **At Beginning:**

#include <iostream>

struct Node {

int data;

Node\* next;

};

// Function to delete the first node from the linked list

void deleteFromStart(Node\*& head) {

if (head == nullptr) {

cout << "The list is already empty. Cannot delete from an empty list." << endl;

return;

}

Node\* temp = head;

head = head->next;

delete temp;

// Deallocate the memory of the removed node }

// Function to display the linked list

void displayList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

} }

int main() {

Node\* head = nullptr; // Initialize an empty linked list

// Insert nodes at the beginning of the list

for (int i = 5; i >= 1; i--) {

Node\* newNode = new Node;

newNode->data = i;

newNode->next = head;

head = newNode;

}

// Display the linked list

cout << "Linked List (Before Deletion): ";

displayList(head);

// Delete the first node

deleteFromStart(head);

// Display the linked list after deletion

cout << "Linked List (After Deletion): ";

displayList(head);

return 0;

}

1. **At End:**

#include <iostream>

struct Node {

int data;

Node\* next;

};

void insertAtEnd(Node\*& head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

} }

// Function to delete the last node from the linked list

void deleteFromEnd(Node\*& head) {

if (head == nullptr) {

cout << "The list is already empty. Cannot delete from an empty list." << endl;

return; }

if (head->next == nullptr) {

delete head;

head = nullptr;

} else {

Node\* current = head;

while (current->next->next != nullptr) {

current = current->next;

}

delete current->next;

current->next = nullptr;

} }

// Function to display the linked list

void displayList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next; }

cout << endl;

}

int main() {

Node\* head = nullptr; // Initialize an empty linked list

// Insert nodes at the end of the list

for (int i = 1; i <= 5; i++) {

insertAtEnd(head, i);

}

// Display the linked list

cout << "Linked List (Before Deletion): ";

displayList(head);

// Delete the last node

deleteFromEnd(head);

// Display the linked list after deletion

cout << "Linked List (After Deletion): ";

displayList(head);

return 0;

}

1. **At Any Point:**

#include <iostream>

struct Node {

int data;

Node\* next;

};

// Function to insert a new node at the end of the linked list

void insertAtEnd(Node\*& head, int value) {

Node\* newNode = new Node;

newNode->data = value;

newNode->next = nullptr;

if (head == nullptr) {

head = newNode;

} else {

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

} }

// Function to delete a node at a specified position in the linked list

void deleteAtPosition(Node\*& head, int position) {

if (head == nullptr) {

cout << "The list is empty. Cannot delete from an empty list." << endl;

return; }

if (position == 0) {

Node\* temp = head;

head = head->next;

delete temp;

} else {

Node\* current = head;

int currentPosition = 0;

while (currentPosition < position - 1 && current->next != nullptr) {

current = current->next;

currentPosition++;

}

if (current->next == nullptr) {

cout << "Invalid position. Node not found." << endl;

} else {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

} } }

// Function to display the linked list

void displayList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

cout << current->data << " ";

current = current->next;

} }

int main() {

Node\* head = nullptr; // Initialize an empty linked list

// Insert nodes at the end of the list

for (int i = 1; i <= 5; i++) {

insertAtEnd(head, i);

}

cout << "Linked List (Before Deletion): ";

displayList(head);

// Delete a node at a specified position

int positionToDelete = 2; // Change this to the desired position

deleteAtPosition(head, positionToDelete);

// Display the linked list after deletion

cout << "Linked List (After Deletion): ";

displayList(head);

return 0;

}