Q-1: Implement a linked list and delete a node with a given value.

Sample test case:

|  |
| --- |
| Input: Enter elements for linked list (enter -1 to stop): 9 5 4 3 -1  Linked List before deletion: 9 5 4 3  Enter a value to delete from the linked list: 5  Output: Linked List after deletion: 9 4 3 |

Solution:

#include <iostream>

using namespace std;

// Node structure for linked list

struct Node {

int data;

Node\* next;

};

// Function to insert an element at the end of the linked list

Node\* insertAtEnd(Node\* head, int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to display the linked list

void displayLinkedList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

}

// Function to delete a node with the given value from the linked list

Node\* deleteNodeWithValue(Node\* head, int value) {

if (head == nullptr) {

return nullptr;

}

if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

return head;

}

Node\* current = head;

while (current->next != nullptr) {

if (current->next->data == value) {

Node\* temp = current->next;

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

delete temp;

break;

}

current = current->next;

}

return head;

}

int main() {

Node\* head = nullptr;

int data;

// Create the linked list with user input

cout << "Enter elements for linked list (enter -1 to stop):\n";

while (true) {

cin >> data;

if (data == -1)

break;

head = insertAtEnd(head, data);

}

// Display the linked list before deletion

cout << "Linked List before deletion: ";

displayLinkedList(head);

int value;

cout << "\nEnter a value to delete from the linked list: ";

cin >> value;

// Delete the node with the given value

head = deleteNodeWithValue(head, value);

// Display the linked list after deletion

cout << "Linked List after deletion: ";

displayLinkedList(head);

// Free memory by deleting nodes

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

Q-2: Implement a sorted linked list and remove duplicates.

Sample test case:

|  |
| --- |
| Input:  Enter elements for sorted linked list (enter -1 to stop):  5 8 12 16 16 17 18 18 -1  Output:  Linked List before removing duplicates: 5 8 12 16 16 17 18 18  Linked List after removing duplicates: 5 8 12 16 17 18 |

Solution:

#include <iostream>

using namespace std;

// Node structure for linked list

struct Node {

int data;

Node\* next;

};

// Function to insert an element at the end of the linked list

Node\* insertAtEnd(Node\* head, int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to display the linked list

void displayLinkedList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

}

// Function to remove duplicates from a sorted linked list

Node\* removeDuplicates(Node\* head) {

if (head == nullptr || head->next == nullptr) {

return head;

}

Node\* current = head;

Node\* nextNode = head->next;

while (nextNode != nullptr) {

if (current->data == nextNode->data) {

Node\* temp = nextNode;

nextNode = nextNode->next;

delete temp;

current->next = nextNode;

} else {

current = nextNode;

nextNode = nextNode->next;

}

}

return head;

}

int main() {

Node\* head = nullptr;

int data;

// Create the linked list with user input

cout << "Enter elements for sorted linked list (enter -1 to stop):\n";

while (true) {

cin >> data;

if (data == -1)

break;

head = insertAtEnd(head, data);

}

// Display the linked list before removing duplicates

cout << "Linked List before removing duplicates: ";

displayLinkedList(head);

// Remove duplicates from the sorted linked list

head = removeDuplicates(head);

// Display the linked list after removing duplicates

cout << "\nLinked List after removing duplicates: ";

displayLinkedList(head);

// Free memory by deleting nodes

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

Q-3: Implement a singly linked list and delete the first occurrence of a given value from the list.

Sample test case:

|  |
| --- |
| Input:  Enter elements for linked list (enter -1 to stop):  3 5 9 12 5 8 -1  Linked List before deleting nodes: 3 5 9 12 5 8  Enter the value to be deleted from the linked list: 5  Output: Linked List after deleting nodes: 3 9 12 5 8 |
| Solution: |

#include <iostream>

using namespace std;

// Node structure for singly linked list

struct Node {

int data;

Node\* next;

};

// Function to insert an element at the end of the linked list

Node\* insertAtEnd(Node\* head, int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to display the linked list

void displayLinkedList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

}

// Function to delete the first occurrence of a given value from the linked list

Node\* deleteFirstNodeWithValue(Node\* head, int value) {

if (head == nullptr) {

return nullptr;

}

if (head->data == value) {

Node\* temp = head;

head = head->next;

delete temp;

return head;

}

Node\* current = head;

Node\* prev = nullptr;

while (current != nullptr) {

if (current->data == value) {

prev->next = current->next;

delete current;

break;

}

prev = current;

current = current->next;

}

return head;

}

int main() {

Node\* head = nullptr;

int data;

// Create the linked list with user input

cout << "Enter elements for linked list (enter -1 to stop):\n";

while (true) {

cin >> data;

if (data == -1)

break;

head = insertAtEnd(head, data);

}

// Display the linked list before deleting nodes

cout << "Linked List before deleting nodes: ";

displayLinkedList(head);

// Prompt the user to enter the value to be deleted

int valueToDelete;

cout << "\nEnter the value to be deleted from the linked list: ";

cin >> valueToDelete;

// Delete the first node with the given value from the linked list

head = deleteFirstNodeWithValue(head, valueToDelete);

// Display the linked list after deleting nodes

cout << "Linked List after deleting nodes: ";

displayLinkedList(head);

// Free memory by deleting nodes

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

Q-4: Implement a singly linked list and delete a node at a given position from the list.

Sample test case:

|  |
| --- |
| Input:  Enter elements for linked list (enter -1 to stop): 8 9 12 4 2 1 -1  Linked List before deleting nodes: 8 9 12 4 2 1  Enter the position of the node to be deleted (0-indexed): 2  Output: Linked List after deleting nodes: 8 9 4 2 1 |

Solution:

#include <iostream>

using namespace std;

// Node structure for singly linked list

struct Node {

int data;

Node\* next;

};

// Function to insert an element at the end of the linked list

Node\* insertAtEnd(Node\* head, int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = nullptr;

if (head == nullptr) {

return newNode;

}

Node\* current = head;

while (current->next != nullptr) {

current = current->next;

}

current->next = newNode;

return head;

}

// Function to display the linked list

void displayLinkedList(Node\* head) {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

}

// Function to delete a node at a given position from the linked list

Node\* deleteNodeAtPosition(Node\* head, int position) {

if (head == nullptr) {

return nullptr;

}

if (position == 0) {

Node\* temp = head;

head = head->next;

delete temp;

return head;

}

Node\* current = head;

Node\* prev = nullptr;

int currentPosition = 0;

while (current != nullptr && currentPosition < position) {

prev = current;

current = current->next;

currentPosition++;

}

if (current == nullptr) {

return head; // Node at the given position not found

}

prev->next = current->next;

delete current;

return head;

}

int main() {

Node\* head = nullptr;

int data;

// Create the linked list with user input

cout << "Enter elements for linked list (enter -1 to stop):\n";

while (true) {

cin >> data;

if (data == -1)

break;

head = insertAtEnd(head, data);

}

// Display the linked list before deleting nodes

cout << "Linked List before deleting nodes: ";

displayLinkedList(head);

// Prompt the user to enter the position of the node to be deleted

int position;

cout << "\nEnter the position of the node to be deleted (0-indexed): ";

cin >> position;

// Delete the node at the given position from the linked list

head = deleteNodeAtPosition(head, position);

// Display the linked list after deleting nodes

cout << "Linked List after deleting nodes: ";

displayLinkedList(head);

// Free memory by deleting nodes

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}