CS-221-L Data Structures and Algorithms Lab

Lab report # ___

Submitted by: Shayan Rizwan Yazdanie

Registration Number: 2024585

Submitted to: Adnan Haider

Semester: 3rd

Faculty of Computer Science and Engineering

GIK Institute of Engineering Sciences and Technology

Task# 01:

Write a function to reverse a singly linked list in-place (without creating a new list). Example: Input List: 10 -> 20 -> 30 -> 40 -> NULL Output List: 40 -> 30 -> 20 -> 10 -> NULL

Solution

CODE:

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

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

int main() {
    cout << "Linked Lists" << endl;

    node* head = NULL;
    node* tail = NULL;

    for (int i = 0; i < 4; i++) {
        node* newnode = new node;
        newnode->data = (i + 1) * 10;
        newnode->next = NULL;

        if (head == NULL) {
            // First node in the list
            head = newnode;
```

```
        tail = newnode;
    } else {
        tail->next = newnode;
        tail = newnode;
    }
}

// Traversing the list
node* temp = head;
cout << "Traversing the list:" << endl;
while (temp != NULL) {
    cout << "Data: " << temp->data << endl;
    cout << "Next: " << temp->next << endl;
    temp = temp->next;
}

// Reversing the list
node* prev = NULL;
node* curr = head;
node* next = NULL;
while (curr != NULL) {
    next = curr->next;
    curr->next = prev;
    prev = curr;
    curr = next;
}
head = prev; // new head of reversed list

// Traversing the reversed list
cout << "Reversed list:" << endl;
temp = head;
```

```
    while (temp != NULL) {
        cout << "Data: " << temp->data << endl;
        temp = temp->next;
    }

    return 0;
}
```

Task# 02:

Find the Middle Element Write a function that finds and prints the middle element of a linked list. Example: Input List: 10 -> 20 -> 30 -> 40 -> 50 -> NULL Output: 30 (the middle node).

Solution

CODE:

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

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

int getLength(node* head) {

   int length = 0;
```

```
    // Count nodes in linked list
    while (head) {
        length++;
        head = head->next;
    }

    return length;
}

int getMiddle(node* head) {

    // finding length of the linked list
    int length = getLength(head);

    // traverse till we reached half of length
    int midIndex = length / 2;
    while (midIndex--) {
        head = head->next;
    }

    return head->data;
}
```

```
int main() {
   cout << "Linked Lists" << endl;


   node* head = NULL;
   node* tail = NULL;


   for (int i = 0; i < 5; i++) {
       node* newnode = new node;
       newnode->data = (i + 1) * 10;
       newnode->next = NULL;


       if (head == NULL) {
           // First node in the list
           head = newnode;
           tail = newnode;
       } else {
           tail->next = newnode;
           tail = newnode;
       }
   }


   // Traversing the list
   node* temp = head;
   cout << "Traversing the list:" << endl;
   while (temp != NULL) {
       cout << "Data: " << temp->data << endl;
```

```
        cout << "Next: " << temp->next << endl;
        temp = temp->next;
    }

    cout << getMiddle(head);

    return 0;
}
```

Task# 03:

Delete Node by Value Write a function to delete the first node containing a given value (not just by position). Example: Input List: 10 -> 20 -> 30 -> 40 -> NULL Delete value = 30 Output List: 10 -> 20 -> 40 -> NULL • Handle special cases: 1. Value is at head. 2. Value not found. 3. Value in the middle/tail.

Solution

CODE:

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

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

// Function to delete the first node containing the given value
```

```
node* deleteNodeByValue(node* head, int value) {
    if (head == NULL) {
        cout << "List is empty. Nothing to delete." << endl;
        return head;
    }

    // Special case: value at head
    if (head->data == value) {
        node* temp = head;
        head = head->next;
        delete temp;
        cout << "Deleted node with value " << value << " from the
     head." << endl;
        return head;
    }

    // For other nodes
    node* current = head;
    while (current->next != NULL && current->next->data != value) {
        current = current->next;
    }

    // Value not found
    if (current->next == NULL) {
        cout << "Value " << value << " not found in the list." <<
     endl;
```

```
        return head;
    }

    // Value found; delete node
    node* temp = current->next;
    current->next = temp->next;
    delete temp;
    cout << "Deleted node with value " << value << "." << endl;
    return head;
}

int main() {
    cout << "Linked Lists" << endl;

    node* head = NULL;
    node* tail = NULL;

    // Creating the list: 10 -> 20 -> 30 -> 40
    for (int i = 0; i < 4; i++) {
        node* newnode = new node;
        newnode->data = (i + 1) * 10;
        newnode->next = NULL;

        if (head == NULL) {
            head = newnode;
            tail = newnode;
```

```
        } else {
            tail->next = newnode;
            tail = newnode;
        }
    }

    // Traversing the list
    node* temp = head;
    cout << "Traversing the list:" << endl;
    while (temp != NULL) {
        cout << "Data: " << temp->data << endl;
        temp = temp->next;
    }

    // Delete a node by value (example: delete 30)
    head = deleteNodeByValue(head, 30);

    // Traversing the list after deletion
    cout << "List after deletion:" << endl;
    temp = head;
    while (temp != NULL) {
        cout << "Data: " << temp->data << endl;
        temp = temp->next;
    }

    return 0;
```

```
}
```