# Data Structures and Algorithms(UCS540)

Sixth-Semester

Submitted by:

Naman Sood [102104012] 3EE2

BE Third Year (2021-2025) Electrical Engineering

Submitted To:

Mr. Yadvendra Singh

Assistant Professor

(Contractual – I)

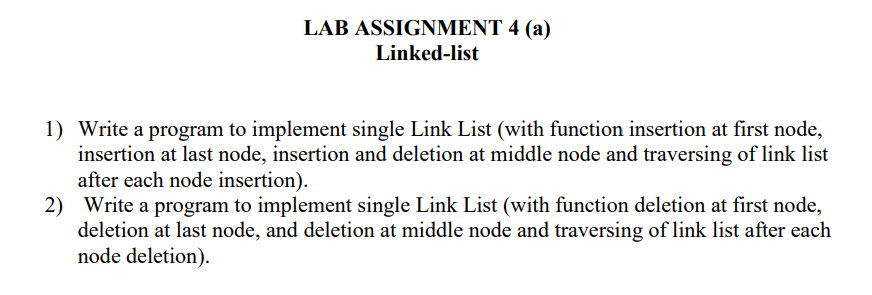


Department of Electrical & Instrumentation Engineering,

Thapar Institute of Engineering & Technology, Patiala

January-May 2024

# List of Experiments



**Q1.**

#include<iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

next = NULL;

}

};

Node\* takeInput() {

int data;

cin >> data;

Node\* head = NULL;

Node\* tail = NULL;

int count = 0;

while (data != -1) {

Node\* newNode = new Node(data);

if (head == NULL) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = tail->next;

}

cin >> data;

count++;

}

cout << "Number of elements = " << count << endl;

return head;

}

void print(Node\* head) {

if (head == NULL) {

cout << "Empty" << endl;

return;

}

cout << "Linked List: ";

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

void insert\_pos(Node\*& head, int pos, int element) {

Node\* ele = new Node(element);

if (pos == 0) {

ele->next = head;

head = ele;

print(head);

return;

}

Node\* temp = head;

int i = 1;

while (temp != NULL && i < pos) {

temp = temp->next;

i++;

}

if (temp != NULL) {

ele->next = temp->next;

temp->next = ele;

print(head);

}

else {

cout << "Invalid position" << endl;

}

}

void delete\_pos(Node\*& head, int pos) {

if (head == NULL) {

cout << "Empty" << endl;

return;

}

if (pos == 0) {

Node\* temp = head;

head = head->next;

delete temp;

print(head);

return;

}

Node\* temp = head;

int i = 1;

while (temp != NULL && i < pos) {

temp = temp->next;

i++;

}

if (temp != NULL && temp->next != NULL) {

Node\* temp1 = temp->next;

temp->next = temp1->next;

delete temp1;

print(head);

}

else {

cout << "Invalid position" << endl;

}

}

int main() {

Node\* head = NULL;

// Insertion at first node

cout << "Inserting 10 at first node:" << endl;

insert\_pos(head, 0, 10);

// Insertion at last node

cout << "Inserting 20 at last node:" << endl;

insert\_pos(head, 1, 20);

// Traversing after each insertion

print(head);

// Insertion at middle node

cout << "Inserting 15 at position 1:" << endl;

insert\_pos(head, 1, 15);

// Traversing after insertion at middle node

print(head);

// Deletion at middle node

cout << "Deleting node at position 1:" << endl;

delete\_pos(head, 1);

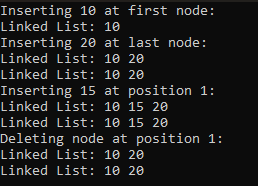
// Traversing after deletion at middle node

print(head);

return 0;

}

**Output:**

****

**Q2.**

#include<iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int data) {

this->data = data;

next = NULL;

}

};

Node\* takeInput() {

int data;

cin >> data;

Node\* head = NULL;

Node\* tail = NULL;

int count = 0;

while (data != -1) {

Node\* newNode = new Node(data);

if (head == NULL) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = tail->next;

}

cin >> data;

count++;

}

cout << "Number of elements = " << count << endl;

return head;

}

void print(Node\* head) {

if (head == NULL) {

cout << "Empty" << endl;

return;

}

cout << "Linked List: ";

Node\* temp = head;

while (temp != NULL) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

void delete\_first(Node\*& head) {

if (head == NULL) {

cout << "Empty list, nothing to delete." << endl;

return;

}

Node\* temp = head;

head = head->next;

delete temp;

print(head);

}

void delete\_last(Node\*& head) {

if (head == NULL) {

cout << "Empty list, nothing to delete." << endl;

return;

}

if (head->next == NULL) {

delete head;

head = NULL;

cout << "List is empty now." << endl;

return;

}

Node\* temp = head;

while (temp->next->next != NULL) {

temp = temp->next;

}

delete temp->next;

temp->next = NULL;

print(head);

}

void delete\_middle(Node\*& head, int pos) {

if (head == NULL) {

cout << "Empty list, nothing to delete." << endl;

return;

}

if (pos == 0) {

delete\_first(head);

return;

}

Node\* temp = head;

int count = 1;

while (count < pos && temp->next != NULL) {

temp = temp->next;

count++;

}

if (temp->next == NULL) {

cout << "Position exceeds the size of the list." << endl;

return;

}

Node\* toDelete = temp->next;

temp->next = temp->next->next;

delete toDelete;

print(head);

}

int main() {

Node\* head = takeInput();

print(head);

// Delete first node

cout << "Deleting first node:" << endl;

delete\_first(head);

// Delete last node

cout << "Deleting last node:" << endl;

delete\_last(head);

// Delete node at position 2 (middle)

cout << "Deleting node at position 2 (middle):" << endl;

delete\_middle(head, 2);

return 0;

}

**Output:**

