**Zainab Shahzad**

**56108**

**DSA Lab 06**

**Task 1:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = nullptr;

return newNode;

}

void deleteFromMid(Node\*\* head, int data) {

if (\*head == nullptr) {

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

return;

}

int length = 0;

Node\* temp = \*head;

do {

temp = temp->next;

length++;

} while (temp != \*head);

int pos = length / 2;

temp = \*head;

for (int i = 0; i < pos - 1; i++) {

temp = temp->next;

}

Node\* nodeToDelete = temp->next;

temp->next = nodeToDelete->next;

delete nodeToDelete;

}

void printList(Node\* head) {

Node\* temp = head;

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

Node\* head = nullptr;

Node\* node1 = createNode(10);

Node\* node2 = createNode(20);

Node\* node3 = createNode(30);

Node\* node4 = createNode(40);

Node\* node5 = createNode(50);

node1->next = node2;

node2->next = node3;

node3->next = node4;

node4->next = node5;

node5->next = node1;

head = node1;

cout << "Original List: ";

printList(head);

deleteFromMid(&head, 30);

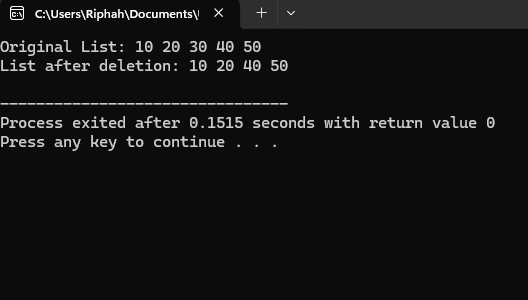
cout << "List after deletion: ";

printList(head);

return 0;

}

**Output:**

****

**TASK 2:**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

Node\* prev;

};

Node\* head = nullptr;

void insertAtStart(int n) {

Node\* newNode = new Node;

newNode->data = n;

if (head == nullptr) {

newNode->next = newNode;

newNode->prev = newNode;

head = newNode;

} else {

Node\* tail = head->prev;

newNode->next = head;

newNode->prev = tail;

head->prev = newNode;

tail->next = newNode;

head = newNode;

}

}

void insertAtEnd(int n) {

Node\* newNode = new Node;

newNode->data = n;

if (head == nullptr) {

newNode->next = newNode;

newNode->prev = newNode;

head = newNode;

} else {

Node\* tail = head->prev;

tail->next = newNode;

newNode->prev = tail;

newNode->next = head;

head->prev = newNode;

}

}

void insertAtPosition(int pos, int n) {

if (pos < 1) {

cout << "Invalid position!" << endl;

return;

}

Node\* newNode = new Node;

newNode->data = n;

if (pos == 1) {

insertAtStart(n);

return;

}

Node\* temp = head;

int count = 1;

while (count < pos - 1 && temp->next != head) {

temp = temp->next;

count++;

}

if (temp->next == head && count < pos - 1) {

cout << "Invalid position!" << endl;

return;

}

newNode->next = temp->next;

newNode->prev = temp;

temp->next->prev = newNode;

temp->next = newNode;

}

void printList() {

if (head == nullptr) {

cout << "List is empty!" << endl;

return;

}

Node\* temp = head;

cout << "List elements: ";

do {

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

temp = temp->next;

} while (temp != head);

cout << endl;

}

int main() {

insertAtStart(3);

insertAtStart(5);

insertAtStart(9);

insertAtStart(8);

printList();

insertAtEnd(7);

printList();

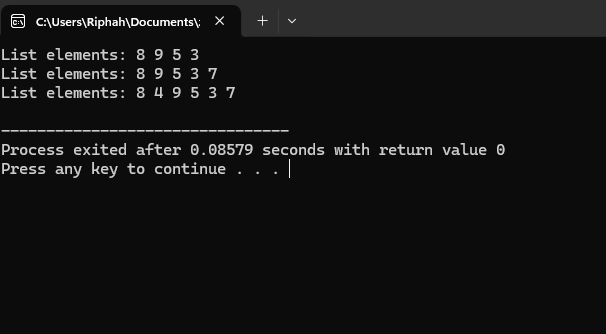
insertAtPosition(2, 4);

printList();

return 0;

}

**OUTPUT:**

****