#include <iostream>

#include <string>

using namespace std;

// Node class

class Node {

public:

string name;

Node\* prev;

Node\* next;

Node(string name) : name(name), prev(nullptr), next(nullptr) {}

};

// DoublyLinkedList class

class DoublyLinkedList {

private:

Node\* head;

Node\* tail;

public:

DoublyLinkedList() : head(nullptr), tail(nullptr) {}

// Task 1: Complete insertAtEnd

void insertAtEnd(string name) {

Node\* newNode = new Node(name);

if (head == nullptr) {

head = tail = newNode;

} else {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

// Task 2: Complete countAndDisplay

void countAndDisplay() {

int count = 0;

Node\* current = head;

while (current != nullptr) {

cout << "Node " << ++count << ": " << current->name << endl;

current = current->next;

}

cout << "Total Nodes: " << count << endl;

}

// Task 3: Complete deleteLastNode

void deleteLastNode() {

if (head == nullptr) {

cout << "The list is empty." << endl;

return;

}

if (head == tail) {

delete head;

head = tail = nullptr;

} else {

Node\* temp = tail;

tail = tail->prev;

tail->next = nullptr;

delete temp;

}

}

// Task 4: Complete insertAtSecond

void insertAtSecond(string name) {

Node\* newNode = new Node(name);

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

cout << "Cannot insert at second position, the list has less than two nodes." << endl;

delete newNode;

return;

}

Node\* second = head->next;

head->next = newNode;

newNode->prev = head;

newNode->next = second;

second->prev = newNode;

}

// Task 5: Complete displayList

void displayList() {

Node\* current = head;

int index = 1;

while (current != nullptr) {

cout << "Node " << index++ << ": " << current->name << endl;

current = current->next;

}

}

};

int main() {

DoublyLinkedList list;

// Step 1: Create a list with names: "Ali", "Baba", "Chan", "Diana", "Ely".

list.insertAtEnd("Ali");

list.insertAtEnd("Baba");

list.insertAtEnd("Chan");

list.insertAtEnd("Diana");

list.insertAtEnd("Ely");

// Step 2: Count and display the nodes.

list.countAndDisplay();

// Step 3: Delete the last node ("Ely") and display the updated list.

cout << "\nList after deleting last node:" << endl;

list.deleteLastNode();

list.displayList();

// Step 4: Insert a new node ("Alisa") at the second position and display the updated list.

cout << "\nList after inserting 'Alisa' at second position:" << endl;

list.insertAtSecond("Alisa");

list.displayList();

return 0;

}