 #include<iostream>

 using namespace std;

 class Node {

 public:

 string data; // Data stored in the node

 Node\* prev; // Pointer to the previous node

 Node\* next; // Pointer to the next node

 // Constructor to initialize a new node

 Node(string data) {

 this->data = data;

 this->prev = nullptr;

 this->next = nullptr;

 }

 };

 class TextEditor {

 private:

 Node\* head; // Pointer to the head of the list

 Node\* tail; // Pointer to the tail of the list

 public:

 TextEditor() { // Constructor to initialize the text editor

 head = nullptr;

 tail = nullptr;

 }

 // Function to insert text at the beginning

 void InsertAtBeg(string data) {

 Node\* newNode = new Node(data);

 if (head == nullptr) { // If the list is empty

 head = tail = newNode;

 return;

 }

 newNode->next = head;

 head->prev = newNode;

 head = newNode;

 }

 // Function to insert text at the end

 void InsertAtEnd(string data) {

Node\* newNode = new Node(data);

 if (head == nullptr) { // If the list is empty

 head = tail = newNode;

 return;

 }

 tail->next = newNode;

 newNode->prev = tail;

 tail = newNode;

 }

 // Function to insert text at a specific position

 void InsertAtPos(string data, int pos) {

 if (pos < 1) {

 cout << "Invalid position!" << endl;

 return;

 }

 if (pos == 1) {

 InsertAtBeg(data);

 return;

 }

 Node\* temp = head;

 for (int i = 1; temp != nullptr && i < pos- 1; i++) {

 temp = temp->next;

 }

 if (temp == nullptr) {

 cout << "Position exceeds list length!" << endl;

 return;

 }

 Node\* newNode = new Node(data);

 newNode->next = temp->next;

 newNode->prev = temp;

 if (temp->next != nullptr) {

 temp->next->prev = newNode;

 } else {

 tail = newNode; // Inserting at the end

 }

 temp->next = newNode;

 }

 // Function to delete the first node

 void DeleteFirst() {

 if (head == nullptr) {

 cout << "List is empty!" << endl;

 return;

}

 Node\* temp = head;

 head = head->next;

 if (head != nullptr) {

 head->prev = nullptr;

 } else {

 tail = nullptr; // List becomes empty

 }

 delete temp;

 }

 // Function to delete the last node

 void DeleteLast() {

 if (tail == nullptr) {

 cout << "List is empty!" << endl;

 return;

 }

 Node\* temp = tail;

 tail = tail->prev;

 if (tail != nullptr) {

 tail->next = nullptr;

 } else {

 head = nullptr; // List becomes empty

 }

 delete temp;

 }

 // Function to delete a node at a specific position

 void DeleteAtPos(int pos) {

 if (head == nullptr) {

 cout << "List is empty!" << endl;

 return;

 }

 if (pos == 1) {

 DeleteFirst();

 return;

 }

 Node\* temp = head;

 for (int i = 1; temp != nullptr && i < pos; i++) {

 temp = temp->next;

 }

 if (temp == nullptr) {

 cout << "Position exceeds list length!" << endl;

 return;

 }

if (temp->next != nullptr) {

 temp->next->prev = temp->prev;

 } else {

 tail = temp->prev; // Deleting the last node

 }

 if (temp->prev != nullptr) {

 temp->prev->next = temp->next;

 }

 delete temp;

 }

 // Function to display the list

 void Display() {

 if (head == nullptr) {

 cout << "List is empty!" << endl;

 return;

 }

 Node\* temp = head;

 while (temp != nullptr) {

 cout << temp->data << "-> ";

 temp = temp->next;

 }

 cout << "NULL" << endl;

 }

 // Function to display the list in reverse order

 void DisplayReverse() {

 if (tail == nullptr) {

 cout << "List is empty!" << endl;

 return;

 }

 Node\* temp = tail;

 while (temp != nullptr) {

 cout << temp->data << "-> ";

 temp = temp->prev;

 }

 cout << "NULL" << endl;

 }

 };

 int main() {

 TextEditor editor; // Create an instance of TextEditor

 string text;

 int choice, pos,n,i;

cout<<"Enter the no of elements";

 cin>>n;

 do {

 cout << "Choose option:" << endl;

 cout << "1. Insert at beginning" << endl;

 cout << "2. Insert at end" << endl;

 cout << "3. Insert at position" << endl;

 cout << "4. Delete first" << endl;

 cout << "5. Delete last" << endl;

 cout << "6. Delete at position" << endl;

 cout << "7. Display list" << endl;

 cout << "8. Display list in reverse" << endl;

 cout << "0. Exit" << endl;

 cin >> choice;

 switch (choice) {

 case 1:

 for(i=0;i<n;i++)

 {

 cout << "Enter text to insert at beginning: ";

 cin >> text;

 editor.InsertAtBeg(text);

 }

 break;

 case 2:

 for(i=0;i<n;i++){

 cout << "Enter text to insert at end: ";

 cin >> text;

 editor.InsertAtEnd(text);}

 break;

 case 3:

 cout << "Enter position: ";

 cin >> pos;

 cout << "Enter text to insert: ";

 cin >> text;

 editor.InsertAtPos(text, pos);

 break;

 case 4:

 editor.DeleteFirst();

 break;

 case 5:

 editor.DeleteLast();

 break;

 case 6:

 cout << "Enter position to delete: ";

 cin >> pos;

 editor.DeleteAtPos(pos);

break;

 case 7:

 editor.Display();

 break;

 case 8:

 editor.DisplayReverse();

 break;

 case 0:

 cout << "Exiting..." << endl;

 break;

 default:

 cout << "Invalid choice!" << endl;

 }

 } while (choice != 0);

 return 0;

 }