Karim Samy Fathy Elmenshawy 231006962

Data Structure

Dr.Ahmed said

Question 1 :

#include <iostream>

using namespace std;

class ListNode {

public:

int value;

ListNode\* next;

ListNode\* prev;

ListNode(int val) : value(val), next(nullptr), prev(nullptr) {}

};

class DoublyLinkedList {

private:

ListNode\* start;

ListNode\* end;

int length;

public:

DoublyLinkedList() : start(nullptr), end(nullptr), length(0) {}

void addNode(int val) {

ListNode\* newNode = new ListNode(val);

if (start == nullptr) {

start = end = newNode;

} else {

end->next = newNode;

newNode->prev = end;

end = newNode;

}

length++;

}

void showList() {

ListNode\* current = start;

while (current != nullptr) {

cout << current->value << " <-> ";

current = current->next;

}

cout << "NULL" << endl;

}

void clearList() {

ListNode\* temp = start;

while (temp != nullptr) {

ListNode\* toDelete = temp;

temp = temp->next;

delete toDelete;

}

start = end = nullptr;

length = 0;

}

~DoublyLinkedList() {

clearList();

}

};

int main() {

DoublyLinkedList myList;

myList.addNode(10);

myList.addNode(20);

myList.addNode(30);

cout << "Doubly Linked List before clearing: " << endl;

myList.showList();

myList.clearList();

cout << "Doubly Linked List after clearing: " << endl;

myList.showList();

return 0;

}

Question 2 :

Singly :

#include <iostream>

using namespace std;

class Element {

public:

int value;

Element\* next;

Element(int val) : value(val), next(nullptr) {}

};

class SortedList {

private:

Element\* head;

public:

SortedList() : head(nullptr) {}

void insertOrdered(int val) {

Element\* newElement = new Element(val);

if (!head || val < head->value) {

newElement->next = head;

head = newElement;

return;

}

Element\* current = head;

while (current->next && current->next->value < val) {

current = current->next;

}

newElement->next = current->next;

current->next = newElement;

}

void showList() {

Element\* current = head;

while (current) {

cout << current->value << " ";

current = current->next;

}

cout << "NULL" << endl;

}

};

int main() {

SortedList list;

list.insertOrdered(40);

list.insertOrdered(15);

list.insertOrdered(30);

list.insertOrdered(5);

list.insertOrdered(25);

cout << "Sorted Singly Linked List: ";

list.showList();

return 0;}

Run :

Sorted Singly Linked List: 5 15 25 30 40 NULL

Question 2 :

Doubly:

#include <iostream>

using namespace std;

class Element {

public:

int value;

Element\* nextNode;

Element\* prevNode;

Element(int val) : value(val), nextNode(nullptr), prevNode(nullptr) {}

};

class SortedDoublyLinkedList {

private:

Element\* headNode;

Element\* tailNode;

int nodeCount;

public:

SortedDoublyLinkedList() : headNode(nullptr), tailNode(nullptr), nodeCount(0) {}

void addElementInSortedOrder(int val) {

Element\* newElement = new Element(val);

if (!headNode || val < headNode->value) {

newElement->nextNode = headNode;

if (headNode) headNode->prevNode = newElement;

headNode = newElement;

if (!tailNode) tailNode = newElement;

return;

}

Element\* current = headNode;

while (current->nextNode && current->nextNode->value < val) {

current = current->nextNode;

}

newElement->nextNode = current->nextNode;

newElement->prevNode = current;

current->nextNode = newElement;

if (newElement->nextNode) {

newElement->nextNode->prevNode = newElement;

} else {

tailNode = newElement;

}

}

void printList() const {

Element\* temp = headNode;

while (temp) {

cout << temp->value << " ";

temp = temp->nextNode;

}

cout << "NULL" << endl;

}

};

int main() {

SortedDoublyLinkedList list;

list.addElementInSortedOrder(30);

list.addElementInSortedOrder(10);

list.addElementInSortedOrder(20);

list.addElementInSortedOrder(5);

list.addElementInSortedOrder(25);

cout << "Sorted Doubly Linked List: ";

list.printList();

return 0;

}

Run:

Sorted Doubly Linked List: 5 10 20 25 30 NULL