Nour Khaled Mohamed

231004419/ Cs6

Bonus assignment

**Question1 (Makenull)**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node\* prev;

Node(int value) : data(value), next(nullptr), prev(nullptr) {}

};

class DoublyList {

private:

Node\* head;

Node\* tail;

int counter;

public:

DoublyList() : head(nullptr), tail(nullptr), counter(0) {}

void insert(int value) {

Node\* newNode = new Node(value);

if (!head) {

head = tail = newNode;

} else {

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

counter++;

}

void display() {

Node\* temp = head;

while (temp) {

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

temp = temp->next;

}

cout << "nullptr" << endl;

}

void makenull() {

while (head) {

Node\* temp = head;

head = head->next;

delete temp;

}

tail = nullptr;

counter = 0;

cout << "List has been reset.\n";

}

};

int main() {

DoublyList list;

list.insert(10);

list.insert(20);

list.insert(30);

cout << "Doubly Linked List before makenull:\n";

list.display();

list.makenull();

cout << "Doubly Linked List after makenull:\n";

list.display();

return 0;

}

**Question2(Insert,Sort singly)**

**}**#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node(int value) : data(value), next(nullptr) {}

};

class SinglyLinkedList {

private:

Node\* head;

public:

SinglyLinkedList() : head(nullptr) {}

void insertSorted(int value) {

Node\* newNode = new Node(value);

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

newNode->next = head;

head = newNode;

return;

}

Node\* current = head;

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

current = current->next;

}

newNode->next = current->next;

current->next = newNode;

}

void display() {

Node\* temp = head;

while (temp) {

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

temp = temp->next;

}

cout << "nullptr" << endl;

}

~SinglyLinkedList() {

while (head) {

Node\* temp = head;

head = head->next;

delete temp;

}

}

};

int main() {

SinglyLinkedList list;

list.insertSorted(30);

list.insertSorted(10);

list.insertSorted(20);

list.insertSorted(5);

list.insertSorted(25);

cout << "Sorted Singly Linked List:\n";

list.display();

return 0;

**Question3(Insert,Sort Doubly)**

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

Node\* prev;

Node(int value) : data(value), next(nullptr), prev(nullptr) {}

};

class DoublyLinkedList {

private:

Node\* head;

Node\* tail;

public:

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

void insertSorted(int value) {

Node\* newNode = new Node(value);

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

newNode->next = head;

if (head) head->prev = newNode;

head = newNode;

if (!tail) tail = newNode;

return;

}

Node\* current = head;

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

current = current->next;

}

newNode->next = current->next;

newNode->prev = current;

current->next = newNode;

if (newNode->next) {

newNode->next->prev = newNode;

} else {

tail = newNode;

}

}

void display() {

Node\* temp = head;

while (temp) {

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

temp = temp->next;

}

cout << "nullptr" << endl;

}

~DoublyLinkedList() {

while (head) {

Node\* temp = head;

head = head->next;

delete temp;

}

tail = nullptr;

}

};

int main() {

DoublyLinkedList list;

list.insertSorted(30);

list.insertSorted(10);

list.insertSorted(20);

list.insertSorted(5);

list.insertSorted(25);

cout << "Sorted Doubly Linked List:\n";

list.display();

return 0;

}