|  |  |
| --- | --- |
| download | COMSATS University Islamabad, Vehari Campus Department of Computer Science |

**Class: BCS-SP22 Submission Deadline: 9 Oct 2023**

**Subject: Data Structures and Algorithms-Lab Instructor: Yasmeen Jana Max Marks: 20 Reg. No: SP22-BCS-006**

**Q#1**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void displayLinkedList(Node\* head) {

Node\* ptr = head;

cout << "The linked list is: "<<endl;

while (ptr !=NULL) {

cout << ptr->data << " ";

ptr = ptr->next;

}

cout << endl;

ptr = head;

cout << "head address: " << &head << endl;

cout << "head content: " << head << endl;

while (ptr != NULL) {

cout << "\*\*ptr address:\* " << &ptr << endl;

cout << "ptr content: " << ptr << endl;

cout << "ptr->data: " << ptr->data << endl;

cout << "ptr: " << ptr << endl;

cout << "ptr->next: " << ptr->next << endl;

ptr = ptr->next;

}

}

int main() {

Node\* head = NULL;

Node\* second = NULL;

Node\* third = NULL;

Node\* fourth = NULL;

head = new Node;

second = new Node;

third = new Node;

fourth = new Node;

head->data = 1;

head->next = second;

second->data = 2;

second->next = third;

third->data = 20;

third->next = fourth;

fourth->data = 30;

fourth->next = NULL;

displayLinkedList(head);

delete head;

delete second;

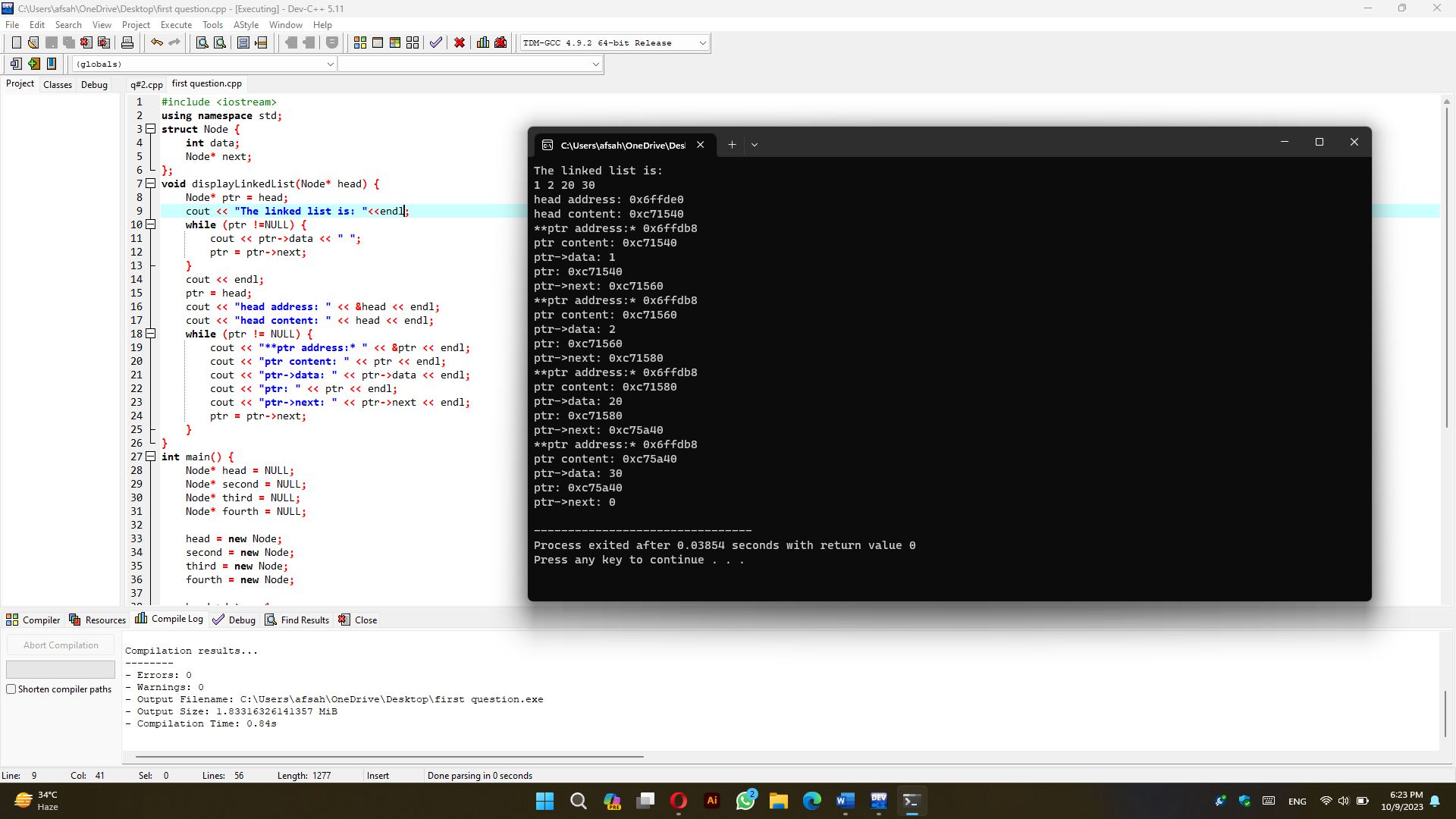
delete third;

delete fourth;

return 0;

}

***OUTPUT:***

******

**Q#2**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

struct DNode {

int data;

DNode\* prev;

DNode\* next;

};

struct CNode {

int data;

CNode\* next;

};

class LinkedList {

public:

LinkedList() {

head = NULL;

tail = NULL;

}

void insertAtBeginning(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = head;

head = newNode;

if (tail == NULL) {

tail = head;

}

}

void insertAtEnd(int data) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

if (tail == NULL) {

head = tail = newNode;

} else {

tail->next = newNode;

tail = newNode;

}

}

void insertAfter(int data, int key) {

Node\* newNode = new Node;

newNode->data = data;

newNode->next = NULL;

if (head == NULL) {

cout << "List is empty. Cannot insert after." << endl;

return;

}

Node\* temp = head;

while (temp != NULL) {

if (temp->data == key) {

newNode->next = temp->next;

temp->next = newNode;

if (temp == tail) {

tail = newNode;

}

return;

}

temp = temp->next;

}

cout << "Key not found in the list." << endl;

}

void deleteNode(int key) {

if (head == NULL) {

cout << "List is empty. Cannot delete." << endl;

return;

}

if (head->data == key) {

Node\* temp = head;

head = head->next;

delete temp;

return;

}

Node\* prev = NULL;

Node\* current = head;

while (current != NULL) {

if (current->data == key) {

prev->next = current->next;

if (current == tail) {

tail = prev;

}

delete current;

return;

}

prev = current;

current = current->next;

}

cout << "Key not found in the list." << endl;

}

void display() {

Node\* temp = head;

while (temp != NULL) {

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

temp = temp->next;

}

cout << endl;

}

void reverse() {

Node\* prev = NULL;

Node\* current = head;

Node\* next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

tail = head;

head = prev;

}

private:

Node\* head;

Node\* tail;

};

int main() {

int choice;

LinkedList singlyLinkedList;

LinkedList doublyLinkedList;

LinkedList circularLinkedList;

do {

cout << "Which linked list you want:" << endl;

cout << "1: Single" << endl;

cout << "2: Double" << endl;

cout << "3: Circular" << endl;

cout << "4: Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int operation;

do {

cout << "Which operation you want to perform:" << endl;

cout << "1: Insertion" << endl;

cout << "2: Deletion" << endl;

cout << "3: Display" << endl;

cout << "4: Reverse" << endl;

cout << "5: Exit" << endl;

cout << "Enter your choice: ";

cin >> operation;

switch (operation) {

case 1: {

int insertChoice;

cout << "1: Insertion at beginning" << endl;

cout << "2: Insertion at end" << endl;

cout << "3: Insertion at specific data node" << endl;

cout << "Enter your choice: ";

cin >> insertChoice;

int data;

cout << "Enter data: ";

cin >> data;

if (insertChoice == 1) {

singlyLinkedList.insertAtBeginning(data);

} else if (insertChoice == 2) {

singlyLinkedList.insertAtEnd(data);

} else if (insertChoice == 3) {

int key;

cout << "Enter key data: ";

cin >> key;

singlyLinkedList.insertAfter(data, key);

} else {

cout << "Invalid choice" << endl;

}

break;

}

case 2: {

int key;

cout << "Enter data to delete: ";

cin >> key;

singlyLinkedList.deleteNode(key);

break;

}

case 3: {

cout << "Single Linked List: ";

singlyLinkedList.display();

break;

}

case 4: {

singlyLinkedList.reverse();

cout << "Single Linked List reversed." << endl;

break;

}

case 5: {

break;

}

default: {

cout << "Invalid choice" << endl;

break;

}

}

} while (operation != 5);

break;

}

case 2: {

break;

}

case 3: {

break;

}

case 4: {

cout << "Exiting program." << endl;

break;

}

default: {

cout << "Invalid choice" << endl;

break;

}

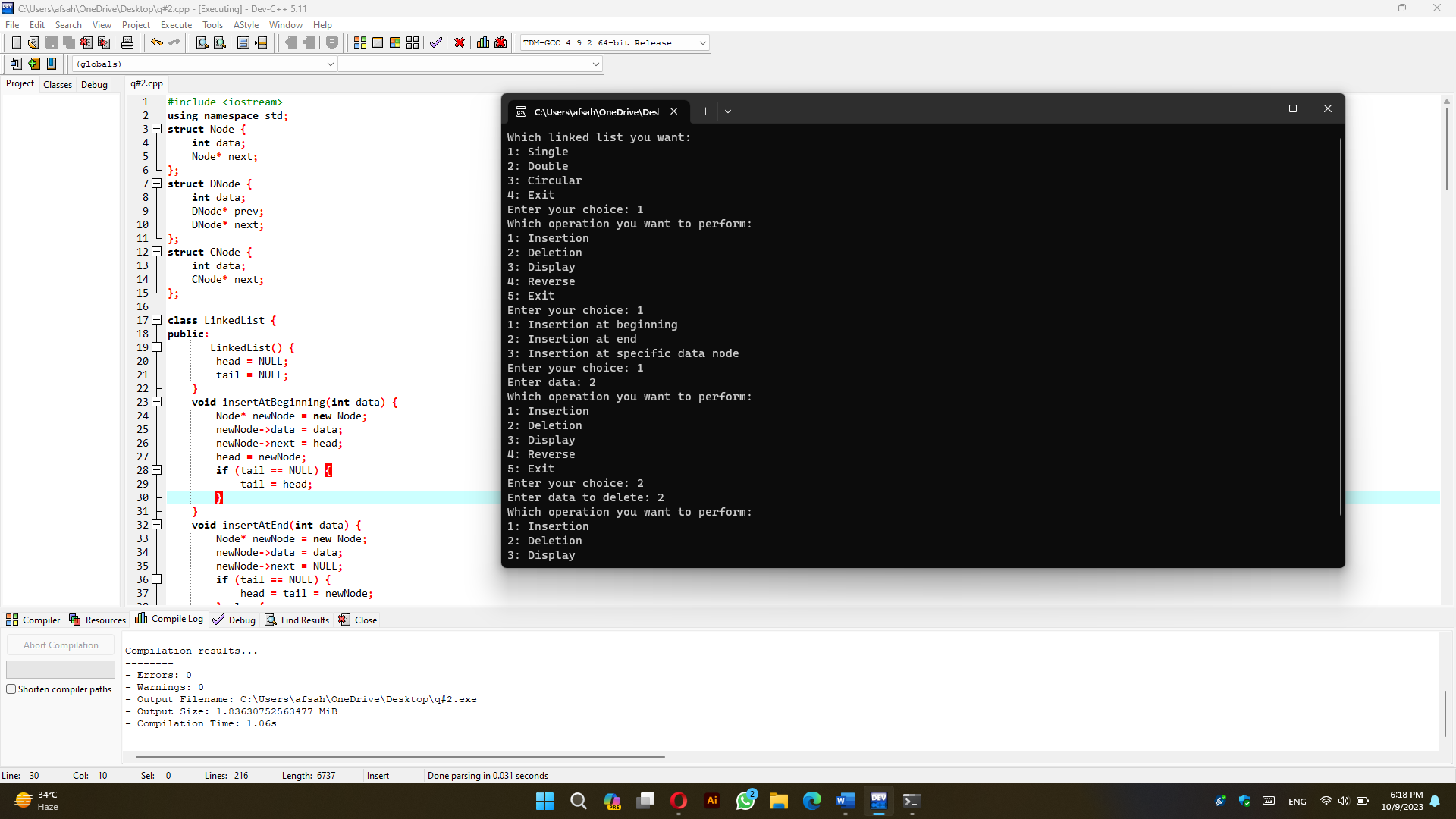
}

} while (choice != 4);

return 0;

}

***OUTPUT:***

******