NAME : RIDHIMA

SUBGROUP:2C32

ROLL NO:1024030466

ASSIGNMENT -5

QUES 1: 1. Develop a menu driven program for the following operations on a Singly Linked List. (a) Insertion at the beginning. (b) Insertion at the end. (c) Insertion in between (before or after a node having a specific value, say 'Insert a new Node 35 before/after the Node 30'). (d) Deletion from the beginning. (e) Deletion from the end. (f) Deletion of a specific node, say 'Delete Node 60'). (g) Search for a node and display its position from head. (h) Display all the node values.

SOLUTION: #include<iostream>

using namespace std;

class node{

    public:

    int data;

    node\* next;

    node(int value){

        data = value;

        next = nullptr;

    }

};

class linkedlist{

    private:

    node\* head;

    public:

    linkedlist(){

        head = nullptr;

    }

    void insertatbegining(int value){

        node\* newnode = new node(value);

        newnode->next = head;

        head = newnode;

    }

    void insertatend(int value){

        node\* newnode = new node(value);

        if(head == nullptr){

            head = newnode;

            return;

        }

        node\* temp = head;

        while(temp->next!=nullptr){

            temp = temp->next;

        }

        temp->next = newnode;

    }

    void insertafter(int prevalue,int value){

        node\* temp = head;

    while (temp != nullptr && temp->data != prevalue) {

        temp = temp->next;

    }

    if (temp == nullptr) {

        cout << "Previous node not found!\n";

        return;

    }

    node\* newNode = new node(value);

    newNode->next = temp->next;

    temp->next = newNode;

    }

    void deleteatbeginning(){

        if (head == nullptr){

            return;

        }

        node\* temp = head;

        head = head->next;

        delete temp;

    }

    void deleteatend(){

        if(head == nullptr){

            return;

        }

        if(head->next == nullptr){

            delete head;

            head = nullptr;

            return;

        }

        node\* temp = head;

        while(temp->next->next!= nullptr){

            temp = temp->next;

        }

        delete temp->next;

        temp->next = nullptr;

    }

    void deletebyvalue(int value){

        if(head == nullptr){

            return;

        }

        if(head->data == value){

            node\* temp = head;

            head = head->next;

            delete temp;

            return;

        }

        node\* temp = head;

        node\* prevnode = nullptr;

        while(temp!=nullptr && temp->data!= value){

            prevnode = temp;

            temp = temp->next;

        }

        if(temp == nullptr){

            return;

        }

        prevnode->next = temp->next;

        delete temp;

    }

    int search(int key) {

        node\* temp = head;

        int pos = 1;

        while (temp != nullptr) {

            if (temp->data == key) {

                return pos;

                cout<<"value found at "<<pos;

            }

            temp = temp->next;

            pos++;

        }

        return -1;

    }

    void printlist(){

        node\* temp = head;

        while(temp!=nullptr){

            cout<<temp->data<<" -> ";

            temp = temp->next;

        }

    }

};

int main(){

    linkedlist l;

    int choice;

    do{

    cout<<"\n==MENU DRIVEN==\n";

    cout<<"Option 1: Insertion at the beginning\n";

    cout<<"Option 2: Insertion at the end\n";

    cout<<"Option 3: Insertion in between\n";

    cout<<"Option 4: Deletion from the beginning.\n";

    cout<<"Option 5: Deletion from the end.\n";

    cout<<"Option 6: Deletion of a specific node\n";

    cout<<"Option 7: Search in linked list .\n";

    cout<<"Option 8: Display all the node values.\n";

    cout<<"Option 9: Exit\n";

    cout<<"Enter your choice:";

    cin>>choice;

    switch(choice){

        case 1 :

        int value;

        cout<<"Value: ";

        cin>>value;

        l.insertatbegining(value);

        break;

        case 2 :

        int value1;

        cout<<"Value: ";

        cin>>value1;

        l.insertatend(value1);

        break;

        case 3 :

        int value5,value2;

        cout<<"Value: ";

        cin>>value5;

        cout<<" prev node value: ";

        cin>>value2;

        l.insertafter(value2,value5);

        break;

        case 4 :

        l.deleteatbeginning();

        break;

        case 5:

        l.deleteatend();

        break;

        case 6:

        int value4;

        cout<<"Value to be deleted";

        cin>>value4;

        l.deletebyvalue(value4);

        break;

        case 7:

        int value3;

        cout<<"Value to be searched :";

        cin>>value3;

        l.search(value3);

        break;

        case 8 :

        l.printlist();

        break;

        case 9:

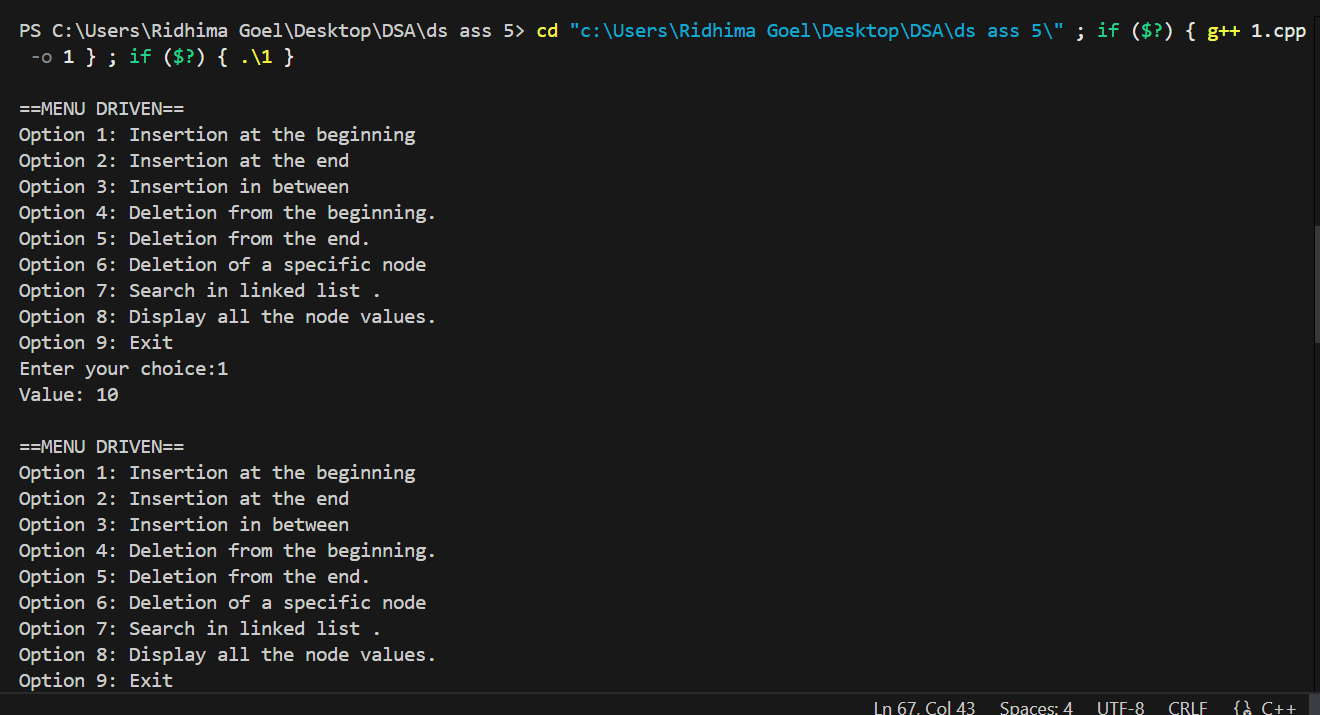
        cout<<"Exit";

    }

    }while(choice!=9);

}

Output:

A screenshot of a computer program

AI-generated content may be incorrect.A black background with white text

AI-generated content may be incorrect.A black rectangle with white text

AI-generated content may be incorrect.

Ques 2:Write a program to count the number of occurrences of a given key in a singly linked list and then delete all the occurrences. Input: Linked List : 1->2->1->2->1->3->1 , key: 1 Output: Count: 4 , Updated Linked List: 2->2->3.

Solution:

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

    Node(int val) {

        data = val;

        next = NULL;

    }

};

// Function to insert node at end

void insert(Node\*& head, int val) {

    Node\* newNode = new Node(val);

    if (head == NULL) {

        head = newNode;

        return;

    }

    Node\* temp = head;

    while (temp->next != NULL) {

        temp = temp->next;

    }

    temp->next = newNode;

}

// Function to display linked list

void display(Node\* head) {

    Node\* temp = head;

    while (temp != NULL) {

        cout << temp->data;

        if (temp->next != NULL) cout << "->";

        temp = temp->next;

    }

    cout << endl;

}

// Function to count occurrences and delete them

int countAndDelete(Node\*& head, int key) {

    int count = 0;

    // Handle head nodes having the key

    while (head != NULL && head->data == key) {

        Node\* toDelete = head;

        head = head->next;

        delete toDelete;

        count++;

    }

    Node\* current = head;

    Node\* prev = NULL;

    while (current != NULL) {

        if (current->data == key) {

            count++;

            prev->next = current->next;

            delete current;

            current = prev->next;

        } else {

            prev = current;

            current = current->next;

        }

    }

    return count;

}

// Driver Code

int main() {

    Node\* head = NULL;

    // Creating linked list: 1->2->1->2->1->3->1

    insert(head, 1);

    insert(head, 2);

    insert(head, 1);

    insert(head, 2);

    insert(head, 1);

    insert(head, 3);

    insert(head, 1);

    cout << "Original Linked List: ";

    display(head);

    int key = 1;

    int count = countAndDelete(head, key);

    cout << "Count of " << key << ": " << count << endl;

    cout << "Updated Linked List: ";

    display(head);

    return 0;

}

Output:

A computer screen with blue and yellow text

AI-generated content may be incorrect.

Ques 3 : 3. Write a program to find the middle of a linked list. Input: 1->2->3->4->5 Output: 3

Solution:

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node\* next;

    Node(int val) : data(val), next(NULL) {}

};

// Insert at end

void insert(Node\*& head, int val) {

    Node\* newNode = new Node(val);

    if (!head) {

        head = newNode;

        return;

    }

    Node\* temp = head;

    while (temp->next) temp = temp->next;

    temp->next = newNode;

}

// Find middle node

int findMiddle(Node\* head) {

    Node\* slow = head;

    Node\* fast = head;

    while (fast && fast->next) {

        slow = slow->next;       // move 1 step

        fast = fast->next->next; // move 2 steps

    }

    return slow->data;

}

// Driver code

int main() {

    Node\* head = NULL;

    // Linked List: 1->2->3->4->5

    insert(head, 1);

    insert(head, 2);

    insert(head, 3);

    insert(head, 4);

    insert(head, 5);

    cout << "Middle element: " << findMiddle(head) << endl;

    return 0;

}

s

solution:

