

**DSA (Data Structure and Algorithms) Lab**

**LAB REPORT # 5**

**Semester**: 3rdSemester

**Section**: C

**Submitted To:**

**Abdullah Shahrose**

**Submitted By:**

**Name**: Faisal Khan

**Roll No**: 22-CS-039

**Task 1:**

#include <iostream>

using namespace std;

class Node

{

public:

    int data;

    Node \*next;

    Node(int val)

    {

        data = val;

        next = nullptr;

    }

};

class LinkedList

{

private:

    Node \*head;

public:

    LinkedList()

    {

        head = nullptr;

    }

    void insertAtStart(int data)

    {

        Node \*newNode = new Node(data);

        newNode->next = head;

        head = newNode;

        cout << "Inserted " << data << " at the start." << endl;

    }

    void insertAtNthPosition(int data, int position)

    {

        if (position <= 0)

        {

            cout << "Invalid position. Please enter a positive position." << endl;

            return;

        }

        Node \*newNode = new Node(data);

        if (position == 1)

        {

            newNode->next = head;

            head = newNode;

            cout << "Inserted " << data << " at position " << position << "." << endl;

            return;

        }

        Node \*current = head;

        int count = 1;

        while (current != nullptr && count < position - 1)

        {

            current = current->next;

            count++;

        }

        if (current == nullptr)

        {

            cout << "Position " << position << " exceeds the length of the list." << endl;

        }

        else

        {

            newNode->next = current->next;

            current->next = newNode;

            cout << "Inserted " << data << " at position " << position << "." << endl;

        }

    }

    void deleteAtStart()

    {

        if (head == nullptr)

        {

            cout << "List is empty. Cannot delete at the start." << endl;

        }

        else

        {

            Node \*temp = head;

            head = head->next;

            cout << "Deleted " << temp->data << " from the start." << endl;

            delete temp;

        }

    }

    void deleteAtNthPosition(int position)

    {

        if (head == nullptr)

        {

            cout << "List is empty. Cannot delete at the specified position." << endl;

            return;

        }

        if (position <= 0)

        {

            cout << "Invalid position. Please enter a positive position." << endl;

            return;

        }

        if (position == 1)

        {

            Node \*temp = head;

            head = head->next;

            cout << "Deleted " << temp->data << " from position " << position << "." << endl;

            delete temp;

            return;

        }

        Node \*current = head;

        int count = 1;

        while (current != nullptr && count < position - 1)

        {

            current = current->next;

            count++;

        }

        if (current == nullptr || current->next == nullptr)

        {

            cout << "Position " << position << " exceeds the length of the list." << endl;

        }

        else

        {

            Node \*temp = current->next;

            current->next = temp->next;

            cout << "Deleted " << temp->data << " from position " << position << "." << endl;

            delete temp;

        }

    }

    void display()

    {

        Node \*current = head;

        cout << "Linked List: ";

        while (current != nullptr)

        {

            cout << current->data << " -> ";

            current = current->next;

        }

        cout << "NULL" << endl;

    }

    ~LinkedList()

    {

        while (head != nullptr)

        {

            Node \*temp = head;

            head = head->next;

            delete temp;

        }

    }

};

int main()

{

    LinkedList list;

    int choice, data, position;

    while (true)

    {

        cout << "---------------------- MAIN MENU ----------------------" << endl;

        cout << "1. Insertion at the start" << endl;

        cout << "2. Insertion at the nth position" << endl;

        cout << "3. Deletion at the start" << endl;

        cout << "4. Deletion at the nth position" << endl;

        cout << "5. Quit" << endl;

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            cout << "Enter data to insert at the start: ";

            cin >> data;

            list.insertAtStart(data);

            list.display();

            break;

        case 2:

            cout << "Enter data to insert: ";

            cin >> data;

            cout << "Enter the position: ";

            cin >> position;

            list.insertAtNthPosition(data, position);

            list.display();

            break;

        case 3:

            list.deleteAtStart();

            list.display();

            break;

        case 4:

            cout << "Enter the position to delete: ";

            cin >> position;

            list.deleteAtNthPosition(position);

            list.display();

            break;

        case 5:

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

            exit(0);

        default:

            cout << "Invalid choice. Please try again." << endl;

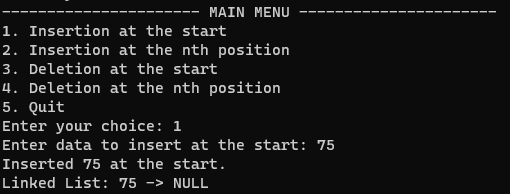
        }

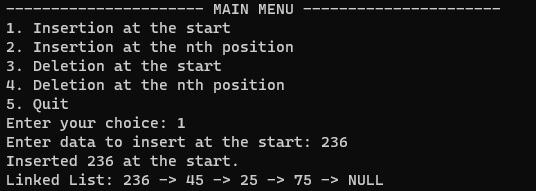
    }

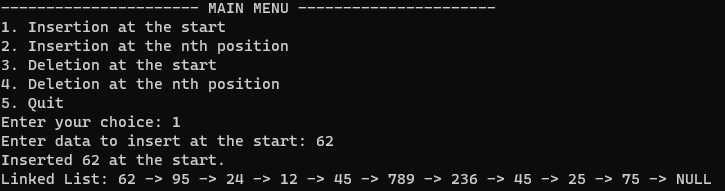
    return 0;

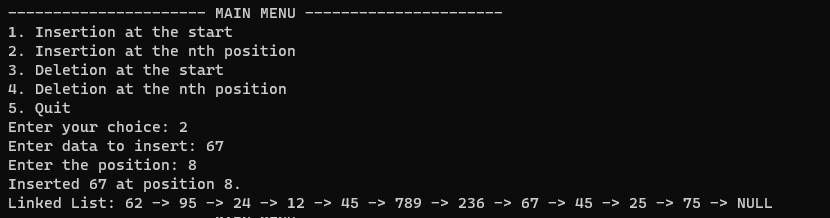
}

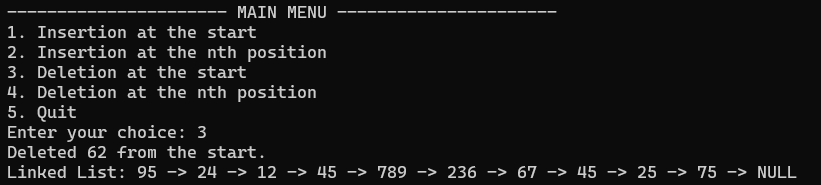
**Output:**

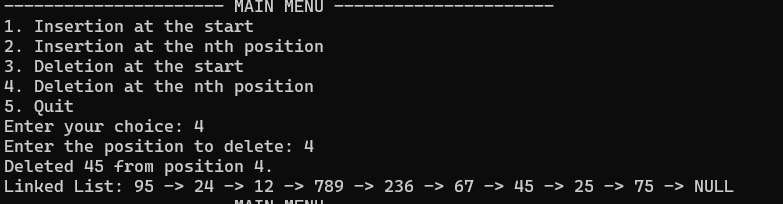
****

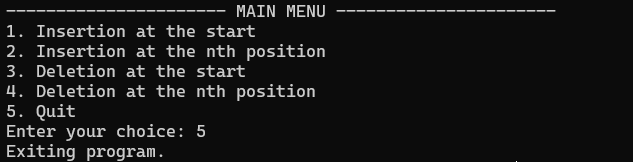
****

****

****

****

****

****