**Experiment 1.3**

**Student Name: Vivek Branch: BE-CSE**

**UID: Section/Group:**

**Date of performance: 25/08/2022 Subject name: Data Structures**

**AIM:**

**Write a menu driven program that maintains a linear linked list whose elements are stored in on ascending order and implements the following operations (using separate functions):**

**a) Insert a new element**

**b) Delete an existing element**

**c) Search an element**

**d) Display all the elements**

**CODE:**

#include <iostream>

using namespace std;

struct Node

{

    int data;

    struct Node \*next;

};

*// Function to print Linked list*

void printList(Node \**node*)

{

    while (*node* != NULL)

    {

        cout << " " << *node*->data;

*node* = *node*->next;

    }

}

*// Function to traverse the Linked list*

void linkedlisttraverse(struct Node \**ptr*)

{

    while (*ptr* != NULL)

    {

        cout << "Element: " << *ptr*->data;

        cout << "\n";

*ptr* = *ptr*->next;

    }

}

*// Function to insert at first in the linked list*

struct Node \*insertatfirst(struct Node \**head*, int *data*)

{

    struct Node \*ptr = (struct Node \*)malloc(sizeof(struct Node));

    ptr->next = *head*;

    ptr->data = *data*;

    return ptr;

}

*// Function to insert at any index in the Linked list*

struct Node \*insertatindex(struct Node \**head*, int *data*, int *index*)

{

    struct Node \*ptr = (struct Node \*)malloc(sizeof(struct Node));

    struct Node \*p = *head*;

    int i = 0;

    while (i != *index* - 1)

    {

        p = p->next;

        i++;

    }

    ptr->data = *data*;

    ptr->next = p->next;

    p->next = ptr;

    return *head*;

}

*// Function to insert at the end of Linked list*

struct Node \*insertAtEnd(struct Node \**head*, int *data*)

{

    struct Node \*ptr = (struct Node \*)malloc(sizeof(struct Node));

    ptr->data = *data*;

    struct Node \*p = *head*;

    while (p->next != NULL)

    {

        p = p->next;

    }

    p->next = ptr;

    ptr->next = NULL;

    return *head*;

}

*// Function to delete at first*

struct Node \*deleteatfirst(struct Node \**head*)

{

    struct Node \*ptr = *head*;

*head* = *head*->next;

    free(ptr);

    return *head*;

}

*// function to delete at any index*

struct Node \*deleteatindex(struct Node \**head*, int *index*)

{

    struct Node \*p = *head*;

    struct Node \*q = *head*->next;

    for (int i = 0; i < *index* - i; i++)

    {

        p = p->next;

        q = q->next;

    }

    p->next = q->next;

    free(q);

    return *head*;

}

*// Function to delete at the end*

struct Node \*deleteatend(struct Node \**head*)

{

    struct Node \*end = *head*;

    struct Node \*prev = NULL;

    while (end->next)

    {

        prev = end;

        end = end->next;

    }

    prev->next = NULL;

    free(end);

    return *head*;

}

*// Searching in Linked List*

void push(Node \*\**head\_ref*, int *new\_key*)

{

    Node \*new\_node = new Node();

    new\_node->data = *new\_key*;

    new\_node->next = (\**head\_ref*);

    (\**head\_ref*) = new\_node;

}

*/\* Checks whether the value x is present in linked list \*/*

bool search(Node \**head*, int *x*)

{

    Node \*current = *head*;

    while (current != NULL)

    {

        if (current->data == *x*)

            return true;

        current = current->next;

    }

    return false;

}

int main()

{

    bool exit = false;

    char YesNo;

    while (!exit)

    {

        struct Node \*head;

        struct Node \*second;

        struct Node \*third;

        struct Node \*fourth;

*// Allocate memory for nodes in the linked list in heap*

        head = (struct Node \*)malloc(sizeof(struct Node));

        second = (struct Node \*)malloc(sizeof(struct Node));

        third = (struct Node \*)malloc(sizeof(struct Node));

        fourth = (struct Node \*)malloc(sizeof(struct Node));

*// link first and second node*

        head->data = 7;

        head->next = second;

*// Link second and third node*

        second->data = 11;

        second->next = third;

*// Link second and thirth node*

        third->data = 25;

        third->next = fourth;

*// terminate the list at fourth node*

        fourth->data = 66;

        fourth->next = NULL;

*// Printing all the operations performed by function calling*

        cout << "\nLinked list after traversing: " << endl;

        linkedlisttraverse(head);

        cout << endl;

        int n;

        cout << "1. Insert a new Element\n";

        cout << "2. Delete an existing element\n";

        cout << "3. Search an Element\n";

        cout << "4. Display all Elements\n";

        cout << "Enter Between 1-4: ";

        cin >> n;

        cout << endl;

        switch (n)

        {

        case 1:

            cout << "Linked list before Insertion: " << endl;

            printList(head);

            cout << endl;

            cout << "\nPerforming Insertion Operations....\n\n";

            cout << "\nLinked list after insertion at first: " << endl;

            head = insertatfirst(head, 56);

            printList(head);

            cout << endl;

            cout << "\nLinked list after insertion at any index: " << endl;

            insertatindex(head, 1, 2);

            printList(head);

            cout << endl;

            cout << "\nLinked list after insertion at end: " << endl;

            head = insertAtEnd(head, 25);

            printList(head);

            cout << endl;

            break;

        case 2:

            cout << "Linked list before Deletion: " << endl;

            printList(head);

            cout << endl;

            cout << "\nPerforming Deletion Operations....\n\n";

            cout << "\nLinked list after deletion at first: " << endl;

            head = deleteatfirst(head);

            printList(head);

            cout << endl;

            cout << "\nLinked list after deletion at any index: " << endl;

            head = deleteatindex(head, 2);

            printList(head);

            cout << endl;

            cout << "\nLinked list after deletion at end: " << endl;

            head = deleteatend(head);

            printList(head);

            cout << endl;

            break;

        case 3:

            cout << "\nPerforming Searching....\n\n";

            search(head, 66) ? cout << "\nElement is present in the Linked List\n" : cout << "\nElement is not present in the Linked List\n";

            break;

        case 4:

            cout << "\nDisplaying Array....\n\n";

            cout << "\nLinked list after all operations: " << endl;

            printList(head);

            break;

        default:

            cout << "\nInvalid Input!!!";

            break;

        }

*//Wish to continue or not*

        cout << " \n \n Do you want to continue? (Y or N) \n";

        cin >> YesNo;

        if (YesNo == 'N' || YesNo == 'n')

        {

            exit = true;

        }

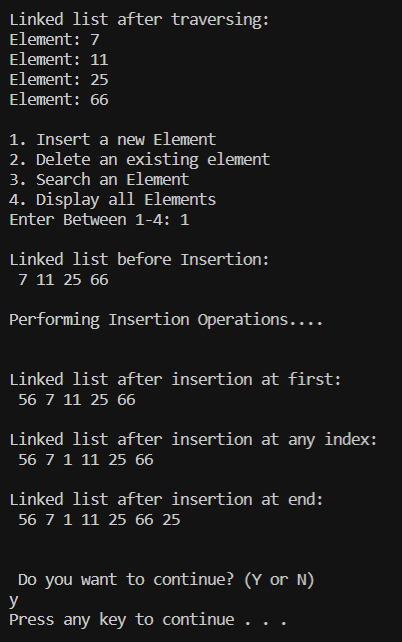
        system("pause");

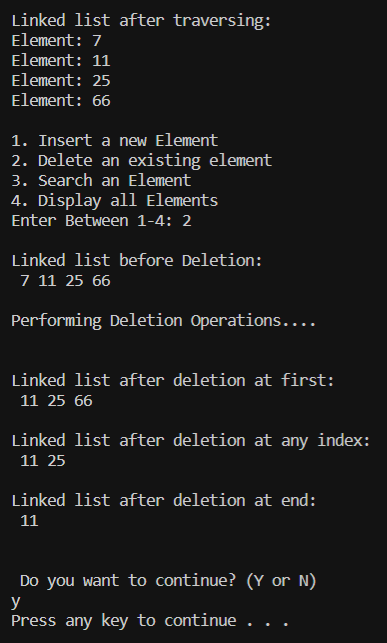
    }

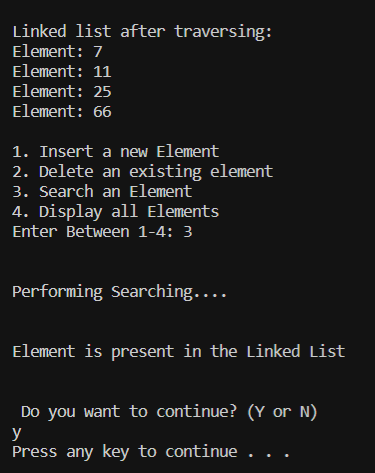
    return 0;

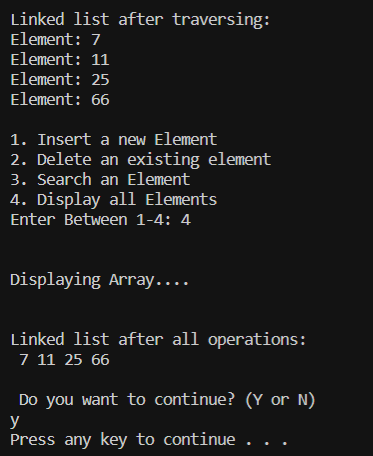
}

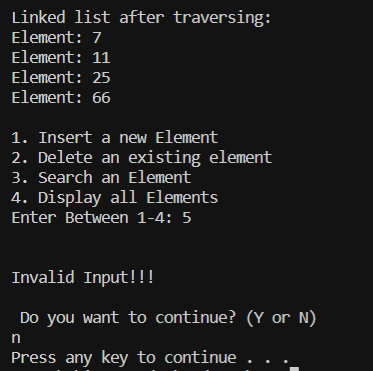
**OUTPUT:**

****

****

****

****

****

Learning outcomes:

1. Learned Singly Linked List
2. Learned about Insertion, Deletion and Searching in Linked List.
3. Learned use of Functions for different operations.
4. Learned concepts of NULL and head in Linked List.

Evaluation Grid:

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. | Student Performance  (Conduct of experiment) objectives/Outcomes. |  | 12 |
| 2. | Viva Voce |  | 10 |
| 3. | Submission of Work Sheet (Record) |  | 8 |
|  | Total |  | 30 |