Date : 10.02.2025
Lakshana Baskaran
24011103026

LAB - 4
List ADT - Singly Linked List

QUESTION 1:

A. Write a C++ menu-driven program to implement List ADT using a singly linked list. Maintain proper boundary conditions and follow good coding practices. The List ADT has the following operations:

1. Insert Beginning
2. Insert End
3. Insert Position
4. Delete Beginning
5. Delete End
6. Delete Position
7. Search
8. Display
9. Display Reverse
10. Reverse Link
11. Exit

SOURCE CODE:

```
//Singly Linked List using list ADT
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
using namespace std;

//class definition for SLL
class node {
    private:
        int data;
        struct node *next;

    public:
        void insert_beginning(int);
```

```
        void insert_end(int);
        void insert_position(int, int);

        int delete_beginning();
        int delete_end();
        int delete_position(int);

        bool search(int);

        void reverse_display(struct node*);
        node* reverse(node*);

        int len();
        void print();

} *head = NULL;

//method to insert element x at the start of the SLL
void node::insert_beginning(int x) {
    struct node* newnode = (struct node*) malloc (sizeof(struct
node));
    newnode -> data = x;
    newnode -> next = head;
    head = newnode;
}

//method to insert element x at the end of the SLL
void node::insert_end(int x) {
    struct node* newnode = (struct node*) malloc (sizeof(struct
node));
    newnode -> data = x;
    newnode -> next = NULL;

    if (head == NULL) {
        head = newnode;
        return;
    }

    struct node* temp = head;
    for (; temp -> next != NULL; temp = temp -> next) {
    }
    temp -> next = newnode;
```

```
}

//method to insert element x at the specified position in the SLL
void node::insert_position(int x, int pos) {
    struct node *newnode = (struct node*) malloc (sizeof(struct
node));

    if (pos > len() || pos < 1) {
        printf("Invalid Postion.\n");
        return;
    }

    if (pos == 1) {
        insert_beginning(x);
        return;
    }

    newnode -> data = x;
    struct node* temp = head;
    for (int i = 1; (temp -> next != NULL) && i < pos-1; i++) {
        temp = temp -> next;
    }
    newnode -> next = temp -> next;
    temp -> next = newnode;
}

//method to delete the first element of the SLL
int node::delete_beginning() {
    if (head == NULL) {
        printf("UnderFlow Error: List is Empty.\n");
        return 0;
    }

    int elem;
    elem = head -> data;
    head = head -> next;

    return elem;
}

//method to delete last element of the SLL
int node::delete_end() {
    if (head == NULL) {
```

```
        printf("UnderFlow Error: List is Empty.\n");
        return 0;
    }

    int elem;
    if (head -> next == NULL) {
        elem = head -> data;
        head = NULL;
        return elem;
    }

    struct node* temp = head;
    for (;temp -> next -> next != NULL; temp = temp -> next) {

    }

    elem = temp -> next -> data;
    temp -> next = NULL;

    return elem;
}

//method to delete the element present in the specified position
int node::delete_position(int pos) {
    if (head == NULL) {
        printf("\nUnderFlow Error: List is Empty.\n");
        return 0;
    }

    if (pos > len() || pos < 0) {
        printf("\nInvalid Position.\n");
        return 0;
    }
    int elem;

    struct node* temp = head;
    if (pos == 1) {
        elem = head -> data;
        head = head -> next;
        return elem;
    }
    for (int i = 1; temp -> next != NULL && i < pos-1; i++) {
        temp = temp -> next;
```

```
    }

    elem = temp -> next -> data;
    temp -> next = temp -> next -> next;

    return elem;
}

//method to check if element x is presentin the SLL or not
bool node::search(int x) {
    struct node* temp = head;
    for (int i = 0; temp -> next != NULL; temp = temp -> next){
        if (temp -> data == x) {
            return 1;
        }
        i++;
    }
    if (temp -> data == x) {
        return 1;
    }

    return 0;
}

//method to display the reverse of the SLL
void node::reverse_display(struct node* temp) {
    if (temp -> next == NULL) {
        printf("%d ", temp -> data);
        return;
    }

    reverse_display(temp->next);
    printf("<- %d ", temp->data);
}

//method to reverse the SLL
node* node::reverse(node *head) {
    node *curr = head, *P = NULL, *N;

    while (curr != NULL) {
        N = curr->next;
        curr->next = P;
```

```
        P = curr;
        curr = N;
    }

    return P;
}

//method to find the number of elements in the SLL
int node::len() {
    int len = 0;
    struct node* temp = head;
    for (; temp -> next != NULL; temp = temp -> next) {
        len++;
    }

    return len+1;
}

//method to display the singly linked list
void node::print() {
    struct node *temp = head;
    if (temp == NULL) {
        printf("head -> NULL\n");
        return;
    }

    //printf("head -> ");
    for (; temp -> next != NULL; temp = temp -> next) {
        printf("%d -> ", temp -> data);
    }
    printf("%d -> NULL\n", temp -> data);
}


int main() {
    node L;
    int x, pos, choice = 0;
    printf("MENU\n1 - Insert Beginning\n2 - Insert At End\n3 - Insert
At Position\n\n");
    printf("4 - delete Beginning\n5 - delete At End\n6 - delete At
Position\n\n");
    printf("7 - Search\n8 - Reverse\n9 - Display\n\n10 - Exit\n\n");
```

Date : 10.02.2025
Lakshana Baskaran
24011103026

```
    while (choice != 10) {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter Element to be inserted: ");
                scanf("%d", &x);
                L.insert_beginning(x);
                break;

            case 2:
                printf("Enter Element to be inserted: ");
                scanf("%d", &x);
                L.insert_end(x);
                break;

            case 3:
                printf("Enter Element to be inserted and its postion:
");
                scanf("%d", &x);
                scanf("%d", &pos);
                L.insert_position(x, pos);
                break;

            case 4:
                printf("%d\n", L.delete_beginning());
                break;

            case 5:
                printf("%d\n", L.delete_end());
                break;

            case 6:
                printf("Enter the postion for deletion: ");
                scanf("%d", &pos);
                printf("%d \n", L.delete_position(pos));
                break;

            case 7:
                printf("Enter Element to be searched: ");
                scanf("%d", &x);

                if (L.search(x) == 1) {
```

```
                    printf("Element Found.\n");
                }
                else {
                    printf("Element Not Found.\n");
                }
                break;

            case 8:
                printf("Reverse of the list: ");
                L.reverse_display(head);
                printf("\n");
                break;

            case 9:
                head = L.reverse(head);
                break;

            case 10:
                printf("Exiting...\n");
                break;

            default:
                //printf("\nInvalid choice. Enter again.\n");
                break;
        }
        if (choice != 10) {
            if (head == NULL) {
                printf("NULL");
            }
            else {
                printf("\nThe list : ");
                L.print();
            }
        }
    }
}
```

Date : 10.02.2025
Lakshana Baskaran
24011103026

OUTPUT:

```
lemon@jupiter:~/workspace/college/DSA/Lab-4$ g++ -o out sll.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-4$ ./out
MENU
1 - Insert Beginning
2 - Insert At End
3 - Insert At Position

4 - delete Beginning
5 - delete At End
6 - delete At Position

7 - Search
8 - Reverse
9 - Display

10 - Exit


Enter your choice: 1
Enter Element to be inserted: 2

The list : 2 -> NULL

Enter your choice: 1
Enter Element to be inserted: 3

The list : 3 -> 2 -> NULL

Enter your choice: 1
Enter Element to be inserted: 4

The list : 4 -> 3 -> 2 -> NULL

Enter your choice: 8
Reverse of the list: 2 <- 3 <- 4

The list : 4 -> 3 -> 2 -> NULL

Enter your choice: 9

The list : 2 -> 3 -> 4 -> NULL
```

Date : 10.02.2025
Lakshana Baskaran
24011103026

```
Enter your choice: 7
Enter Element to be searched: 4
Element Found.

The list : 2 -> 3 -> 4 -> NULL

Enter your choice: 7
Enter Element to be searched: 9
Element Not Found.

The list : 2 -> 3 -> 4 -> NULL

Enter your choice: 6
Enter the postion for deletion: 2
3

The list : 2 -> 4 -> NULL

Enter your choice: 5
4

The list : 2 -> NULL

Enter your choice: 4
2
NULL
Enter your choice: 10
Exiting...
lemon@jupiter:~/workspace/college/DSA/Lab-4$
```

Date : 10.02.2025
Lakshana Baskaran
24011103026

QUESTION 2:

Write a C++ menu-driven program to implement List ADT using a singly linked list.
You have a gethead() private member function that returns the address of the head value of a list. Maintain proper boundary conditions and follow good coding practices. The List ADT has the following operations:

1. Insert Ascending
2. Merge
3. Display
4. Exit

Option 1 inserts a node so the list is always in ascending order.
Option 2 takes two lists as input and merges them into a third list. The third list should also be in ascending order.
Convert the file into a header file and include it in a C++ file.
The second C++ program consists of 3 lists and has the following operations:

1. Insert List1
2. Insert List2
3. Merge into List3
4. Display
5. Exit

SOURCE CODE:

```
#include<iostream>
#include<stdio.h>
using namespace std;
#include "list.h"

int main(){
    List l1;
    int choice;
    int number1, number2, value;
    do  {
        cout << "\nMENU:" << endl;
        cout << "1. Insert " << endl;
        cout << "2. Merge the two lists " << endl;
        cout << "3. Display " << endl;
        cout << "4. Exit" << endl;

        cout << "Enter your choice from the above menu: " << endl;
```

```
        cin >> choice;

        switch(choice) {
                case 1:    //This function enters the lists
                cout << "Enter the number of elements to add in list1
(in descending order): ";
                cin >> number1;
                        for(int i=0 ;i < number1 ;i++)
                        {
                            printf("Enter %d element: ", i+1);
                            scanf("%d", &value);
                            l1.insert_beginning(
l1.gethead1(),value);
                        }
                            cout << "List 1: ";
                            l1.display(l1.gethead1());

                cout << "Enter the number of elements you want to add
in list2 (in descending order): " ;
                cin >> number2;
                        for(int i=0 ; i< number2 ; i++){
                            printf("Enter %d element: ", i+1);
                            scanf("%d", &value);
                            l1.insert_beginning(l1.gethead2() ,
value);
                        }
                        cout << "List 2: ";
                        l1.display(l1.gethead2());

                break;

                case 2: // to merge and diplay the list
                    cout << "The function of merging and sorting the
lists in ascending order is performed." << endl;
                    cout << "Click on the Display option to display
the merged list." << endl;
                break;

                case 3:
                cout << "Merged list: " ;
                l1.displaym();
                break;
```

```
                case 4:
                cout << "Exciting the program..." << endl;
                break;
        }
    } while(choice != 4);

   return 0;
}
```

Date : 10.02.2025
Lakshana Baskaran
24011103026

SOURCE CODE:

```
lemon@jupiter:~/workspace/college/DSA/Lab-4$ g++ -o out dsalab.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-4$ ./out

MENU:
1. Insert
2. Merge the two lists
3. Display
4. Exit
Enter your choice from the above menu:
1
Enter the number of elements you want to add in list1 (in descending order): 3
Enter 1 element: 5
Enter 2 element: 3
Enter 3 element: 1
List 1: 1->3->5->NULL
Enter the number of elements you want to add in list2 (in descending order): 3
Enter 1 element: 6
Enter 2 element: 4
Enter 3 element: 2
List 2: 2->4->6->NULL

MENU:
1. Insert
2. Merge the two lists
3. Display
4. Exit
Enter your choice from the above menu:
2
The function of merging and sorting the lists in ascending order is performed.
Click on the Display option to display the merged list.

MENU:
1. Insert
2. Merge the two lists
3. Display
4. Exit
Enter your choice from the above menu:
3
Merged list: 1->2->3->4->5->6->NULL

MENU:
1. Insert
2. Merge the two lists
3. Display
4. Exit
Enter your choice from the above menu:
4
Exciting the program...
lemon@jupiter:~/workspace/college/DSA/Lab-4$
```

—