LAB SUBMISSION – 4

Meher Shrishti Nigam – 20BRS1193

All the required functions are implemented in the following C code.

Output is at the end.

1- Write a program (preferably in C) to insert the element in singly linked list at the following location

a) At beginning consider the case where no elements are present in list and another where elements are already present in the list.

b) At given location (also include the cases for after given location and before given location.

c) at the end

d) Display the total inserted elements in list.

2- Write a program to calculate the total number of elements present in linked list

3- Write a program for the various cases of deletion:

a) delete the element at the end

b) delete the element at first position

c) delete the element at given position by user

4) Write a program where you have to insert the element in a singly linked list after one given item (consider each item appear only once in the list). Perform the deletion operation for the node which appears after the given item.

```c
// SinglyLinkedList.c
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
/*  A singly linked list here is defined by just the "start".
"start" is a node pointer that doesn't store any data, just
stores the address of the first element in the linked list.
"len" stores the number of elements in the linked list. This can be
easily calculated, however it is added to make things easier and  for
illustration. It can be removed.

    Functions are written independent of len.
    Functions provided:

    insertAtStart
    insertAtEnd
    insertAtPosition
    insertAfterPosition
```

```
    insertBeforePosition
    insertAfterGivenData

    display
    count
    search

    deleteAtStart
    deleteAtEnd
    deleteAtPosition
    deleteAfterGivenData
*/

typedef struct Node{
    int data;
    struct Node * next;
}Node;

typedef struct SinglyLinkedList
{
    Node * start;
    int len;
}SinglyLinkedList;

Node * createNode(int item)
{
    Node * temp = (Node *)malloc(sizeof(Node));
    temp->data = item;
    temp->next = NULL;
    return temp;
}

SinglyLinkedList * createSinglyLinkedList()
{
    SinglyLinkedList * sll = (SinglyLinkedList
*)malloc(sizeof(SinglyLinkedList));
    sll->len = 0;

    /* Creating a start node that will store the location of the
first node. It stores null at the time of declaration.  */
    sll->start = (Node *)malloc(sizeof(Node));
    sll->start->data = INT_MIN;
```

```c
// Data in the start node is never to be accessed. If INT_MIN is the
data displayed, an error has possibly occured
    sll->start->next = NULL;
    printf("A new singly linked list was created!\n");
    return sll;
}

/* Insertion */

// Inserts an element at the start of a linked list
void insertAtStart(SinglyLinkedList * sll, int item)
{
    Node * newnode = createNode(item);
    newnode->next = sll->start->next;
    sll->start->next = newnode;
    printf("%d was inserted at the start of the singly linked
list!\n", item);
    sll->len++;
}

// Inserts an element at the end of a linked list
void insertAtEnd(SinglyLinkedList * sll, int item)
{
    Node * newnode = createNode(item);
    Node * ptr = sll->start;
    while(ptr->next != NULL)
        ptr = ptr->next;
    ptr->next = newnode;
    printf("%d was inserted at the end of the singly linked list!\n",
item);
    sll->len++;
}

// Inserts an element at a specified position.
// Here, position is determined by usual 1-base counting. If position
= 5, item will be the fifth element in the linked list
void insertAtPosition(SinglyLinkedList * sll, int item, int position)
{
    if(position > sll->len + 1)
    {
        printf("Invalid Location\n");
        return;
    }
```

```
    Node * newnode = createNode(item);

    Node * ptr = sll->start;
    for(int i = 0; i < (position - 1); i++)
        ptr = ptr->next;
    newnode->next = ptr->next;
    ptr->next = newnode;
    printf("%d was inserted at position %d of the singly linked
list!\n", item, position);
    sll->len++;
}

// Inserts an element after a specified position.
// Here, position is determined by usual 1-base counting. If position
= 5, item will be the sixth element in the linked list.
void insertAfterPosition(SinglyLinkedList * sll, int item, int
position)
{
    insertAtPosition(sll, item, position + 1);
}
// Inserts an element before a specified position.
// Here, position is determined by usual 1-base counting. If position
= 5, item will be the fourth element in the linked list.
void insertBeforePosition(SinglyLinkedList * sll, int item, int
position)
{
    insertAtPosition(sll, item, position - 1);
}

// Insert an element after a given data. Example, in 10 20 30 40,
enter item = 50 after data = 20, to make it 10 20 50 30 40.
void insertAfterGivenData(SinglyLinkedList * sll, int item, int data)
{
    /*
     *   Another method -
     *   int position = search(sll, data);
     *   if(position == INT_MIN)
     *   {
     *      printf("Data doesn't exist\n");
     *      return;
     *   }
     *   insertAtPosition(sll, item, position);
```

```
     */
    if(sll->start == NULL)
    {
        printf("Data doesn't exist\n");
        return;
    }
    Node * ptr = sll->start;

    while(ptr->data != data)
    {
        ptr = ptr->next;
        if(ptr->data != data && ptr->next == NULL)
        {
            printf("Data doesn't exist\n");
            return;
        }
    }
    Node * newnode = createNode(item);
    newnode->next = ptr->next;
    ptr->next = newnode;
    printf("%d was inserted after %d in the singly linked list!\n",
item, data);
    sll->len++;
}

/* Traversal */
/* Display function provided for illustrative purposes.*/
void display(SinglyLinkedList * sll)
{
    int count = 0;
    // Printing data
    Node * ptr = sll->start->next; // Setting pointer to the first
element in the singly linked list
    printf("Start: %d \n", sll->start);
    while(ptr != NULL)
    {
        count++;
        printf("%d      ", ptr->data);
        ptr = ptr->next;
    }
    // Printing the location
    Node * ptr2 = sll->start->next;
    printf("\n%d ", sll->start->next);
```

```
    while(ptr2 != NULL)
    {
        printf("%d ", ptr2->next);
        ptr2 = ptr2->next;
    }
    printf("\nTotal Elements in the Linked List: %d\n\n", count);
}

void count(SinglyLinkedList * sll)
{
    int count = 0;
    Node * ptr = sll->start->next; // Setting pointer to the first
element in the singly linked list
    while(ptr != NULL)
    {
        count++;
        ptr = ptr->next;
    }
    printf("\nTotal Elements in the Linked List: %d", count);
    // Comparing count to len (count stored in singly linked list
data structure)
    if(count == sll->len)
        printf(" which is equal to len.\n");
    else
        printf(" which is not equal to len.\n");
}

// Returns the 1-base position of the item.
int search(SinglyLinkedList * sll, int item)
{
    int count = 1;
    Node * ptr = sll->start->next;
    while(ptr->data != item)
    {
        ptr = ptr->next;
        count++;
        if(ptr->data != item && ptr->next == NULL)
        {
            printf("%d not found in the linked list.\n", item);
            return INT_MIN;
        }
    }
```

```
    printf("%d found at %d position in the singly linked list.\n",
item, count);
    return count;
}

/* Deletion */

// Deletion at start
int deleteAtStart(SinglyLinkedList * sll)
{
    if(sll->start == NULL)
    {
        printf("Singly Linked List is empty, nothing to delete\n");
        return INT_MIN;
    }
    Node * ptr = sll->start->next;
    sll->start->next = ptr->next;
    int data = ptr->data;
    free(ptr);
    printf("%d was deleted from the start of the singly linked
list.\n", data);
    sll->len--;
    return data;
}

// Deletion at end
int deleteAtEnd(SinglyLinkedList * sll)
{
    if(sll->start == NULL)
    {
        printf("Singly Linked List is empty, nothing to delete\n");
        return INT_MIN;
    }
    Node * ptr = sll->start;
    while(ptr->next->next != NULL)
       ptr = ptr->next;
    Node * ptr2 = ptr->next;
    int data = ptr2->data;
    free(ptr2);
    ptr->next = NULL;
    printf("%d was deleted from the end of the singly linked
list.\n", data);
    sll->len--;
```

```
    return data;
}

// Deletion of an element at a particular position
// Here, position is determined by usual 1-base counting.
int deleteAtPosition(SinglyLinkedList * sll, int position)
{
    if(sll->start == NULL)
    {
        printf("Singly Linked List is empty, nothing to delete\n");
        return INT_MIN;
    }

    if(position > sll->len)
    {
        printf("Invalid Location\n");
        return INT_MIN;
    }

    Node * ptr = sll->start;
    for(int i = 0; i < (position - 1); i++)
        ptr = ptr->next;
    Node * ptr2 = ptr->next;
    ptr->next = ptr2->next;
    int data = ptr2->data;
    free(ptr2);
    sll->len--;
    printf("%d was deleted from %d position of the singly linked
list.\n", data, position);
    return data;
}

// Delete the element present after a given data
int deleteAfterGivenData(SinglyLinkedList * sll, int data)
{
    if(sll->start == NULL)
    {
        printf("Singly Linked List is empty, nothing to delete\n");
        return INT_MIN;
    }
    Node * ptr = sll->start;
    while(ptr->data != data)
    {
```

```
        ptr = ptr->next;
        if(ptr->data != data && ptr->next == NULL)
        {
            printf("Data doesn't exist\n");
            return INT_MIN;
        }
    }
    if(ptr->next == NULL)
    {
        printf("This is the last element, nothing to delete after
this.\n");
        return INT_MIN;
    }
    Node * ptr2 = ptr->next;
    ptr->next = ptr2->next;
    int val = ptr2->data;
    free(ptr2);
    printf("The element after %d was %d, and it has been deleted from
the singly linked list.\n", data, val);
    sll->len--;
    return val;
}

int main()
{
    SinglyLinkedList * sll = createSinglyLinkedList();
    // Insertion At Start
    insertAtStart(sll, 10); // Singly Linked List is empty initially
    display(sll);
    insertAtStart(sll, 20); // Singly Linked List has one element
initially
    display(sll);
    insertAtStart(sll, 30);
    display(sll);
    // Insertion at the end
    insertAtEnd(sll, 40);
    display(sll);
    insertAtEnd(sll, 50);
    display(sll);
    insertAtEnd(sll, 60);
    display(sll);
    // Insertion at position
    insertAtPosition(sll, 70, 3);
```

```
    display(sll);
    insertBeforePosition(sll, 80, 2);
    display(sll);
    insertAfterPosition(sll, 90, 6);
    display(sll);
    // Insertion after given data
    insertAfterGivenData(sll, 100, 60);
    display(sll);
    insertAfterGivenData(sll, 110, 90);
    display(sll);
    insertAfterGivenData(sll, 120, 25); // 25 is not present in the
sll
    display(sll);
    // Delete at start
    deleteAtStart(sll);
    display(sll);
    deleteAtStart(sll);
    display(sll);
    // Delete at end
    deleteAtEnd(sll);
    display(sll);
    deleteAtEnd(sll);
    display(sll);
    // Inserting a few more values
    insertAtEnd(sll, 40);
    insertAtEnd(sll, 50);
    insertAtEnd(sll, 60);
    // Delete at position
    deleteAtPosition(sll, 5);
    display(sll);
    deleteAtPosition(sll, 1);
    display(sll);
    deleteAtPosition(sll, 8);
    display(sll);
    // Delete after given data
    deleteAfterGivenData(sll, 120);
    display(sll);
    deleteAfterGivenData(sll, 50);
    display(sll);
    deleteAfterGivenData(sll, 50);
    display(sll);
    deleteAfterGivenData(sll, 50);
    display(sll);
```

```c
    // Count
    count(sll);
    // Search
    search(sll, 50);
    search(sll, 45);
}
```

```
PS C:\Users\meher\OneDrive\Documents\ASem3\DSA\Lab\Lab_4> gcc SinglyLinkedList.c
PS C:\Users\meher\OneDrive\Documents\ASem3\DSA\Lab\Lab_4> ./a.exe
A new singly linked list was created!
10 was inserted at the start of the singly linked list!
Start: 529504
10
529536 0
Total Elements in the Linked List: 1

20 was inserted at the start of the singly linked list!
Start: 529504
20      10
529568 529536 0
Total Elements in the Linked List: 2

30 was inserted at the start of the singly linked list!
Start: 529504
30      20      10
529600 529568 529536 0
Total Elements in the Linked List: 3

40 was inserted at the end of the singly linked list!
Start: 529504
30      20      10      40
529600 529568 529536 529632 0
Total Elements in the Linked List: 4

50 was inserted at the end of the singly linked list!
Start: 529504
30      20      10      40      50
529600 529568 529536 529632 529664 0
Total Elements in the Linked List: 5

60 was inserted at the end of the singly linked list!
Start: 529504
30      20      10      40      50      60
529600 529568 529536 529632 529664 529696 0
Total Elements in the Linked List: 6

70 was inserted at position 3 of the singly linked list!
Start: 529504
30      20      70      10      40      50      60
529600 529568 529728 529536 529632 529664 529696 0
Total Elements in the Linked List: 7

80 was inserted at position 1 of the singly linked list!
Start: 529504
80      30      20      70      10      40      50      60
529760 529600 529568 529728 529536 529632 529664 529696 0
Total Elements in the Linked List: 8
```

```
90 was inserted at position 7 of the singly linked list!
Start: 529504
80      30      20      70      10      40      90      50      60
529760 529600 529568 529728 529536 529632 529792 529664 529696 0
Total Elements in the Linked List: 9

100 was inserted after 60 in the singly linked list!
Start: 529504
80      30      20      70      10      40      90      50      60      100
529760 529600 529568 529728 529536 529632 529792 529664 529696 529824 0
Total Elements in the Linked List: 10

110 was inserted after 90 in the singly linked list!
Start: 529504
80      30      20      70      10      40      90      110      50      60      100
529760 529600 529568 529728 529536 529632 529792 529856 529664 529696 529824 0
Total Elements in the Linked List: 11

Data doesn't exist
Start: 529504
80      30      20      70      10      40      90      110      50      60      100
529760 529600 529568 529728 529536 529632 529792 529856 529664 529696 529824 0
Total Elements in the Linked List: 11

80 was deleted from the start of the singly linked list.
Start: 529504
30      20      70      10      40      90      110      50      60      100
529600 529568 529728 529536 529632 529792 529856 529664 529696 529824 0
Total Elements in the Linked List: 10

30 was deleted from the start of the singly linked list.
Start: 529504
20      70      10      40      90      110      50      60      100
529568 529728 529536 529632 529792 529856 529664 529696 529824 0
Total Elements in the Linked List: 9

100 was deleted from the end of the singly linked list.
Start: 529504
20      70      10      40      90      110      50      60
529568 529728 529536 529632 529792 529856 529664 529696 0
Total Elements in the Linked List: 8

60 was deleted from the end of the singly linked list.
Start: 529504
20      70      10      40      90      110      50
529568 529728 529536 529632 529792 529856 529664 0
Total Elements in the Linked List: 7
```

```
40 was inserted at the end of the singly linked list!
50 was inserted at the end of the singly linked list!
60 was inserted at the end of the singly linked list!
90 was deleted from 5 position of the singly linked list.
Start: 529504
20      70      10      40      110      50      40      50      60
529568 529728 529536 529632 529856 529664 529696 529824 529600 0
Total Elements in the Linked List: 9

20 was deleted from 1 position of the singly linked list.
Start: 529504
70      10      40      110      50      40      50      60
529728 529536 529632 529856 529664 529696 529824 529600 0
Total Elements in the Linked List: 8

60 was deleted from 8 position of the singly linked list.
Start: 529504
70      10      40      110      50      40      50
529728 529536 529632 529856 529664 529696 529824 0
Total Elements in the Linked List: 7

Data doesn't exist
Start: 529504
70      10      40      110      50      40      50
529728 529536 529632 529856 529664 529696 529824 0
Total Elements in the Linked List: 7

The element after 50 was 40, and it has been deleted from the singly linked list.
Start: 529504
70      10      40      110      50      50
529728 529536 529632 529856 529664 529824 0
Total Elements in the Linked List: 6

The element after 50 was 50, and it has been deleted from the singly linked list.
Start: 529504
70      10      40      110      50
529728 529536 529632 529856 529664 0
Total Elements in the Linked List: 5

This is the last element, nothing to delete after this.
Start: 529504
70      10      40      110      50
529728 529536 529632 529856 529664 0
Total Elements in the Linked List: 5


Total Elements in the Linked List: 5 which is equal to len.
50 found at 5 position in the singly linked list.
45 not found in the linked list.
PS C:\Users\meher\OneDrive\Documents\ASem3\DSA\Lab\Lab_4>
```