1.) To implement Singly LL(Searching, Insertion, Deletion)

Code :

```
#include<iostream>

using namespace std;



class Node
{
   public:
   int data;

   Node *next;
};
void push(Node** head_ref, int new_data)
{

   Node* new_node = new Node();

   new_node->data = new_data;

   new_node->next = (*head_ref);

   (*head_ref) = new_node;
}
void printList(Node *node)
{
   while (node != NULL)
   {
       cout<<" "<<node->data;
       node = node->next;
   }
}
bool search(Node* head, int x)
{
   Node* current = head; // Initialize current
```

```
    while (current != NULL)
    {
        if (current->data == x)
            return true;
        current = current->next;
    }
    return false;
}
void deleteNode(struct Node **head_ref, int key)
{
    // Store head node
    struct Node* temp = *head_ref, *prev;

    if (temp != NULL && temp->data == key)
    {
        *head_ref = temp->next;    // Changed head
        free(temp);                // free old head
        return;
    }

    while (temp != NULL && temp->data != key)
    {
        prev = temp;
        temp = temp->next;
    }
     // If key was not present in linked list
    if (temp == NULL) return;
     // Unlink the node from linked list
    prev->next = temp->next;
     free(temp);
}
int main()
{    int n;
    Node* head = NULL;
    cout<<"enter no of elements to insert "<<endl;
    cin>>n;
    int a[n];
    for (int i = 0; i < n; i++)
    {    cout<<"enter the element to insert"<<endl;
        cin>>a[i];
        push(&head,a[i]);
    }
```

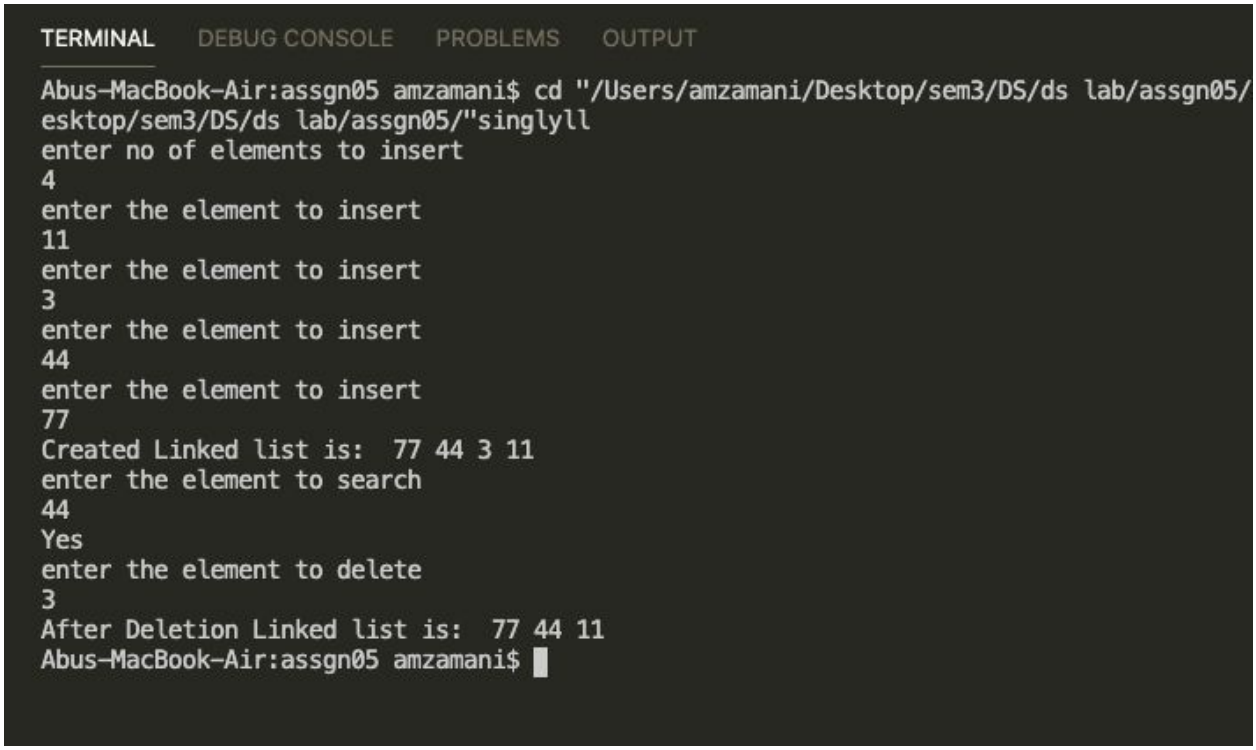
```
    cout<<"Created Linked list is: ";
    printList(head);

    cout<<endl;
    int s;
    cout<<"enter the element to search"<<endl;
    cin>>s;
    search(head, s)? cout<<"Yes" : cout<<"No";
    cout<<endl;
    int d;
    cout<<"enter the element to delete"<<endl;
    cin>>d;
    deleteNode(&head, d);
    cout<<"After Deletion Linked list is: ";
    printList(head);
    cout<<endl;

    return 0;

}
```

Output :


TERMINAL
DEBUG CONSOLE
PROBLEMS
OUTPUT
Abus-MacBook-Air:assgn05 amzamani$ cd "/Users/amzamani/Desktop/sem3/DS/ds lab/assgn05/
esktop/sem3/DS/ds lab/assgn05/"singlyll
enter no of elements to insert
4
enter the element to insert
11
enter the element to insert
3
enter the element to insert
44
enter the element to insert
77
Created Linked list is: 77 44 3 11
enter the element to search
44
Yes
enter the element to delete
3
After Deletion Linked list is: 77 44 11
Abus-MacBook-Air:assgn05 amzamani$

2.) To implement Doubly LL(Searching, Insertion, Deletion)

Code :

```
#include<iostream>

using namespace std;



class Node
{
   public:
   int data;

   Node *next;
   Node *prev;
};

void push(struct Node** head_ref, int new_data)
{

   struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));

   new_node->data = new_data;

   new_node->next = (*head_ref);
   new_node->prev = NULL;

   if ((*head_ref) != NULL)
       (*head_ref)->prev = new_node;

   (*head_ref) = new_node;
}
// void deleteNode(struct Node **head_ref, int key)
// {
//     // Store head node
//     struct Node* temp = *head_ref, *prev;
 //     // If head node itself holds the key to be deleted
```

```
//     if (temp != NULL && temp->data == key)
//     {
//         *head_ref = temp->next;   // Changed head
//         free(temp);               // free old head
//         return;
//     }

//     while (temp != NULL && temp->data != key)
//     {
//         prev = temp;
//         temp = temp->next;
//     }
 //     // If key was not present in linked list
//     if (temp == NULL) return;
 //     // Unlink the node from linked list
//     prev->next = temp->next;
 //     free(temp);
// }

void deleteNode(struct Node** head_ref, struct Node* del)
{
   /* base case */
   if (*head_ref == NULL || del == NULL)
       return;

   if (*head_ref == del)
       *head_ref = del->next;

   if (del->next != NULL)
       del->next->prev = del->prev;

   if (del->prev != NULL)
       del->prev->next = del->next;

   free(del);
   return;
}
void printList(struct Node* node)
{
   struct Node* last;
   printf("\nTraversal in forward direction \n");
   while (node != NULL) {
```

```
        printf(" %d ", node->data);
        last = node;
        node = node->next;
    }
     printf("\nTraversal in reverse direction \n");
    while (last != NULL) {
        printf(" %d ", last->data);
        last = last->prev;
    }
}
bool search(Node* head, int x)
{
    Node* current = head; // Initialize current
    while (current != NULL)
    {
        if (current->data == x)
            return true;
        current = current->next;
    }
    return false;
}
int main()
{
    /* Start with the empty list */
    // struct Node* head = NULL;

    // push(&head, 7);
     // push(&head, 1);
     // push(&head, 4);
     // Insert 8, before 1. So linked list becomes 4->8->1->7->NULL

    int n;
    Node* head = NULL;
    cout<<"enter no of elements to insert "<<endl;
    cin>>n;
    int a[n];
    for (int i = 0; i < n; i++)
    {    cout<<"enter the element to insert"<<endl;
        cin>>a[i];
        push(&head,a[i]);
    }
```

```
   int d;
   cout<<"Created Doubly Linked list is: ";
   printList(head);
   cout<<endl;
   int s;
   cout<<"enter the element to search"<<endl;
   cin>>s;
   search(head, s)? cout<<"Yes" : cout<<"No";
   cout<<endl;
   cout<<"enter the element to delete"<<endl;
   cin>>d;
   deleteNode(&head, head->next);
   cout<<"After Deletion Linked list is: ";
   printList(head);
   cout<<endl;
    cout<<endl;

   getchar();
   return 0;
}
```

Output :

```
TERMINAL    DEBUG CONSOLE    PROBLEMS    OUTPUT


Abus-MacBook-Air:assgn05 amzamani$ cd "/Users/amzamani/Desktop/sem3/DS/ds lab/assgn05/"
esktop/sem3/DS/ds lab/assgn05/"doublyll
enter no of elements to insert
3
enter the element to insert
11
enter the element to insert
22
enter the element to insert
33
Created Doubly Linked list is:
Traversal in forward direction
 33  22  11
Traversal in reverse direction
 11  22  33
enter the element to search
11
Yes
enter the element to delete
22
After Deletion Linked list is:
Traversal in forward direction
 33  11
Traversal in reverse direction
 11  33
```

3.) To implement Circular LL(Searching, Insertion, Deletion)

Code :

```
#include<iostream>
using namespace std;
 struct Node
{
   int data;
   struct Node *next;
};
 struct Node *addToEmpty(struct Node *last, int data)
{
   // This function is only for empty list
   if (last != NULL)
     return last;
    // Creating a node dynamically.
   struct Node *temp =
          (struct Node*)malloc(sizeof(struct Node));
    // Assigning the data.
   temp -> data = data;
   last = temp;
    // Creating the link.
   last -> next = last;
    return last;
}
 struct Node *addBegin(struct Node *last, int data)
{
   if (last == NULL)
       return addToEmpty(last, data);
    struct Node *temp =
           (struct Node *)malloc(sizeof(struct Node));
    temp -> data = data;
   temp -> next = last -> next;
   last -> next = temp;
    return last;
}
 struct Node *addEnd(struct Node *last, int data)
{
```

```
   if (last == NULL)
       return addToEmpty(last, data);

   struct Node *temp =
       (struct Node *)malloc(sizeof(struct Node));
    temp -> data = data;
   temp -> next = last -> next;
   last -> next = temp;
   last = temp;
    return last;
}
 struct Node *addAfter(struct Node *last, int data, int item)
{
   if (last == NULL)
       return NULL;
    struct Node *temp, *p;
   p = last -> next;
   do
   {
       if (p ->data == item)
       {
           temp = (struct Node *)malloc(sizeof(struct Node));
           temp -> data = data;
           temp -> next = p -> next;
           p -> next = temp;
            if (p == last)
               last = temp;
           return last;
       }
       p = p -> next;
   }  while(p != last -> next);
    cout << item << " not present in the list." << endl;
   return last;
 }
bool search(Node* head, int x)
{
   Node* current = head; // Initialize current
   while (current != NULL)
   {
       if (current->data == x)
           return true;
       current = current->next;
```

```
    }
    return false;
}
 void traverse(struct Node *last)
{
    struct Node *p;
     // If list is empty, return.
    if (last == NULL)
    {
        cout << "List is empty." << endl;
        return;
    }
     // Pointing to first Node of the list.
    p = last -> next;
     // Traversing the list.
    do
    {
        cout << p -> data << " ";
        p = p -> next;
     }
    while(p != last->next);
 }
void deleteNode(Node** head, int key)
{

    // If linked list is empty
    if (*head == NULL)
        return;

    // If the list contains only a single node
    if((*head)->data==key && (*head)->next==*head)
    {
        free(*head);
        *head=NULL;
        return;
    }

    Node *last=*head,*d;

    // If head is to be deleted
    if((*head)->data==key) {
```

```
        // Find the last node of the list
        while(last->next!=*head)
            last=last->next;

        // Point last node to the next of head i.e.
        // the second node of the list
        last->next=(*head)->next;
        free(*head);
        *head=last->next;
    }


    while(last->next!=*head&&last->next->data!=key) {
        last=last->next;
    }


    if(last->next->data==key) {
        d=last->next;
        last->next=d->next;
        free(d);
    }
    else
        cout<<"no such keyfound";
    }

int main()
{
    struct Node *last = NULL;
    // int n;
    // cout<<"enter no of elements to insert in empty circular ll "<<endl;
    // cin>>n;
    // int a[n];
    // for (int i = 0; i < n; i++)
    // {   cout<<"enter the element to insert"<<endl;
    //     cin>>a[i];
    //     last = addToEmpty(last,a[i]);
    // }
    // traverse(last);
    // int b;
    // cout<<"enter element to insert at begin"<<endl;
    // cin>>b;
```

```
    // cout<<endl;
    // last = addBegin(last,b);
    // int l;
    // cout<<"enter element to insert at last"<<endl;
    // cin>>l;
    // last = addEnd(last,l );

    // cout<<"enter the elemetn";

    int s;
    last = addToEmpty(last, 6);
    traverse(last);
    cout<<endl;
    last = addBegin(last, 4);
    last = addBegin(last, 2);
    traverse(last);
    cout<<endl;
    last = addEnd(last, 8);
    last = addEnd(last, 12);
    traverse(last);
    cout<<endl;
    last = addAfter(last, 10, 8);
    cout<<endl;
    traverse(last);
    cout<<endl;
    cout<<"enter the element to search"<<endl;
    cin>>s;
    search(last, s)? cout<<"Yes" : cout<<"No";
    cout<<endl;
    int d;
    cout<<"enter the element to delete"<<endl;
    cin>>d;
    deleteNode(&last, d);
    cout<<"After Deletion Linked list is: ";

    traverse(last);
    cout<<endl;
    return 0;
}
```

Output :

```
196     last = addToEmpty(last, 6);
197     traverse(last);
198     cout<<endl;
199     last = addBegin(last, 4);
200     last = addBegin(last, 2);
201     traverse(last);
202     cout<<endl;
203     last = addEnd(last, 8);
204     last = addEnd(last, 12);
205     traverse(last);
206     cout<<endl;
207     last = addAfter(last, 10, 8);
208     cout<<endl;
209     traverse(last);
```

TERMINAL DEBUG CONSOLE PROBLEMS OUTPUT

```
Abus-MacBook-Air:assgn05 amzamani$ cd "/Users/amzamani/Desktop/sem3/DS/ds lab/assgn05/"
ni/Desktop/sem3/DS/ds lab/assgn05/"circularll
6
2 4 6
2 4 6 8 12

2 4 6 8 10 12
enter the element to search
8
Yes
enter the element to delete
10
After Deletion Linked list is: 2 4 6 8 12
Abus-MacBook-Air:assgn05 amzamani$
```