EXPERIMENT – 1.3

Name: Rohan Jaiswal **UID:** 21BCS2856

Branch: CSE **Section/Group:** 608 (B)

Semester: 3rd **Date of Performance:** 1st sept

Subject Name: DS **Subject Code:** 21CSH-211

Aim of Practical:

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

Program Code:

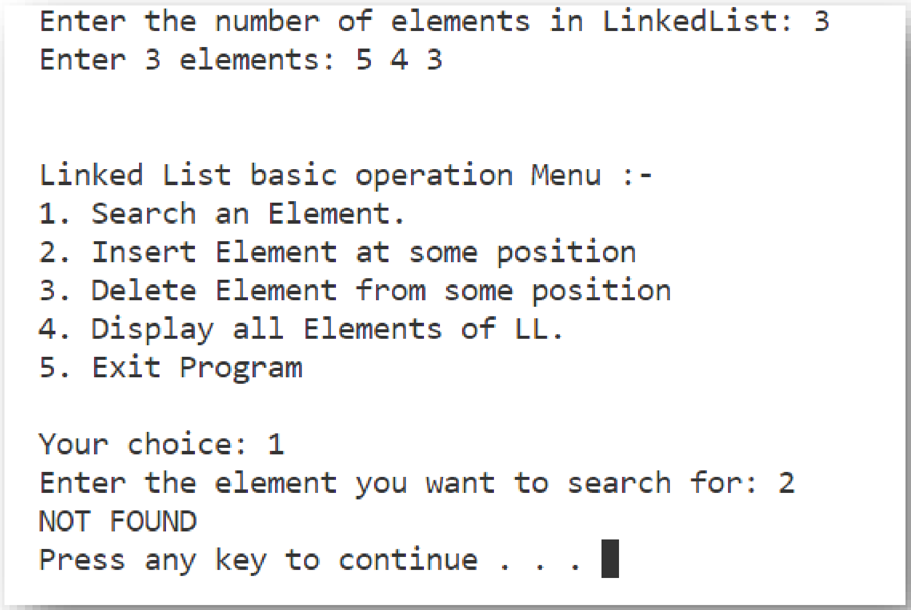
```
#include<bits/stdc++.h>
using ll = long long;

struct Node{
    int data;
    struct Node* next;
    Node(int val){ //constructor
        data = val;
        next = NULL;
    }
};
```

```cpp
void display(Node* head){
    Node* itr = head;
    std::cout<<"\n";
    while(itr){
        std::cout<<itr->data<<" ";
        itr = itr->next;
    }
}

int search(Node* head, int k){
    Node* itr = head;
    int cnt =0;
    while(itr){
        cnt++;
        if(itr->data == k)
            return cnt;
        itr = itr->next;
    }
    return 0;
}

bool insert(Node** head, int pos, int val){
    Node* tmp = new Node(val);

    if(pos==1){
        tmp->next=*head;
        *head = tmp;
        return true;
    }
    else{
        Node* itr = *head;
        int cnt =0;

        while(itr){
            cnt++;
            if(cnt==(pos-1)){
                tmp->next = itr->next;
                itr->next = tmp;
                return true;
            }
            itr = itr->next;
        }
    }
    return false;
}

bool del(Node** head, int pos){
    if(*head == NULL){
        std::cout<<"LinkedList is empty!\n";
        return false;
    }
    if(pos==1){
        Node* tmp = *head;
        *head = tmp->next;
        delete tmp;
        return true;
    }
    else{
        Node* itr = *head;
        int cnt =0;

        while(itr){
            cnt++;
            if(cnt == (pos-1)){
                Node* tmp = itr->next;
                itr->next = tmp->next;
                delete tmp;
                return true;
            }
        }
    }
    return false;
}

int main(){
    Node* head=NULL;
    Node* tail=NULL;
    Node* tmp;

    std::cout<<"Enter the number of elements in LinkedList: ";
    int n;std::cin>>n;
    std::cout<<"Enter "<<n<<" elements: ";
    while(n--){
```

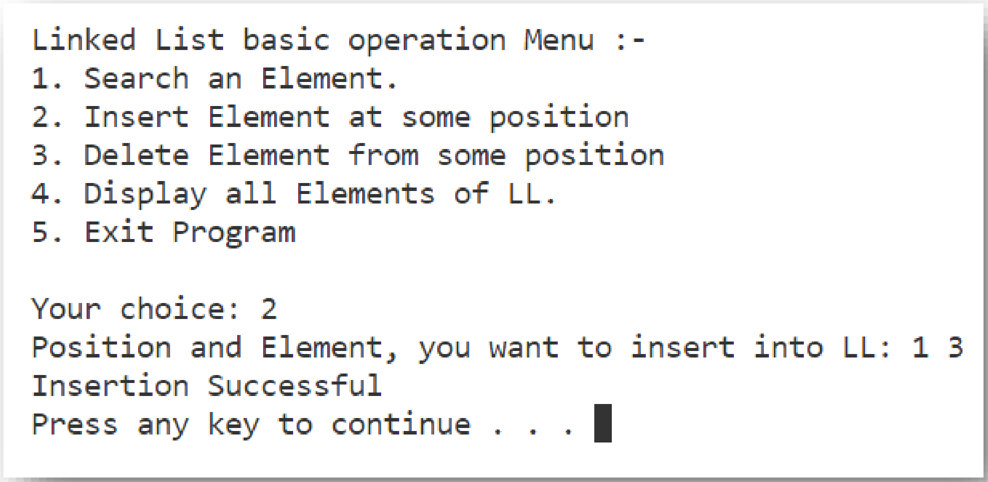
```
        int x;std::cin>>x;
        tmp = new Node(x);

        if(!head){
            head = tmp;
            tail = tmp;
        }
        else{
            tail->next = tmp;
            tail = tail->next;
        }
    }
    bool flag = true;

    while(flag){
        std::cout<<"\n\nLinked List basic operation Menu :-\n";
        std::cout<<"1. Search an Element.\n2. Insert Element at some position\n3. 
Delete Element from some position\n4. Display all Elements of LL.\n5. Exit 
Program\n\n";
        std::cout<<"Your choice: ";
        int res;
        std::string choice; std::cin>>choice;

        if(choice.size()>1)
            choice[0]='6';


        switch (choice[0])
        {
        case '1':
            std::cout<<"Enter the element you want to search for: ";
            int k;std::cin>>k;
            res = search(head, k);
            if(res)
                std::cout<<"Element Found at "<<res<<" position";
            else
                std::cout<<"NOT FOUND";
            break;

        case '2':
            std::cout<<"Position and Element, you want to insert into LL: ";
            int pos,val;std::cin>>pos>>val;
```

```
                if(insert(&head,pos,val))
                    std::cout<<"Insertion Successful";
                else
                    std::cout<<"Insertion Unsuccessful. Position Out of Bound";
                break;

            case '3':
                std::cout<<"Position of Element you want to delete: ";
                int del_pos;std::cin>>del_pos;

                if(del(&head,del_pos)){
                    std::cout<<"Deletion Successful\n";
                }
                else{
                    std::cout<<"Unsuccessful";
                }
                break;

            case '4':
                display(head);
                break;

            case '5':
                flag = false;
                std::cout<<"Exiting.......";
                break;

            default:
                std::cout<<"Invalid Choice... try again!";
                break;
            }
            std::cout<<"\n";
            system("pause");
            std::cout << "\033[2J\033[1;1H"; //for clearing screen in terminal.
        }
        std::cout<<"Program Stopped!!";
    }
```

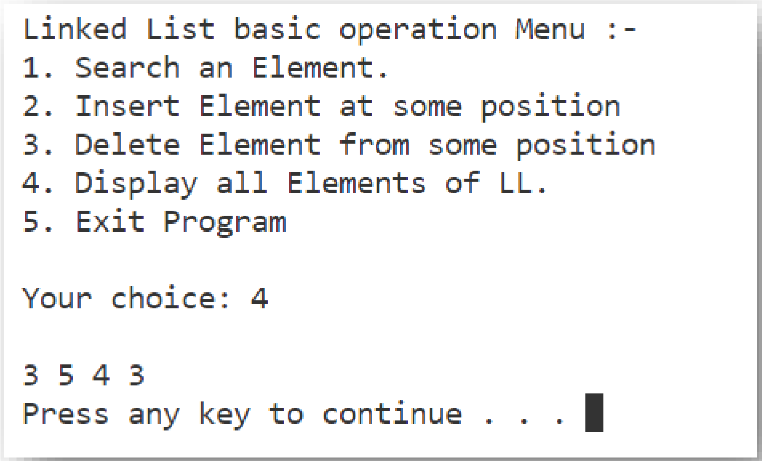
Output:



```
Enter the number of elements in LinkedList: 3
Enter 3 elements: 5 4 3


Linked List basic operation Menu :-
1. Search an Element.
2. Insert Element at some position
3. Delete Element from some position
4. Display all Elements of LL.
5. Exit Program

Your choice: 1
Enter the element you want to search for: 2
NOT FOUND
Press any key to continue . . .
```



```
Linked List basic operation Menu :-
1. Search an Element.
2. Insert Element at some position
3. Delete Element from some position
4. Display all Elements of LL.
5. Exit Program

Your choice: 2
Position and Element, you want to insert into LL: 1 3
Insertion Successful
Press any key to continue . . .
```



```
Linked List basic operation Menu :-
1. Search an Element.
2. Insert Element at some position
3. Delete Element from some position
4. Display all Elements of LL.
5. Exit Program

Your choice: 4

3 5 4 3
Press any key to continue . . .
```