|  |
| --- |
| Experiment No. 7 |
| Program for data structure using built in function for link list, stack and queues |
| Date of Performace: 28/02/2024 |
| Date of Submission: 13/03/2024 |

**Experiment No. 7**





          







**Code:**

#Creating an Empty list

lst = []

#Number of elements in a list

n = int(input("Enter number of elements : "))

print("The elements are : ")

for i in range(0, n):

element = int(input())

#Append the elements

lst.append(element)

print("List : ",lst)

#Traversaling the element

print("The elements in the list are : ")

for element in lst:

print(element)

#Insert the element

a = int(input("Enter the element to insert : "))

b = int(input("Enter the index to insert at : "))

lst.insert(b,a)

print("After insertion : ",lst)

#Remove the element

c = int(input("Enter the element to remove : "))

lst.remove(c)

print("After removing : ",lst)

#Search location of element

d = int(input("Enter the element to search : "))

if(lst.index(d)):

print("Element is present at index : ",lst.index(d))

else:

print("Element is not present")

#Replace the Element

e = int(input("Enter the element to remove : "))

lst.remove(e)

f = int(input("Enter the element to insert : "))

g = int(input("Enter the index to insert at : "))

lst.insert(g,f)

print("After replacing : ",lst)

#Size of the list

print("The size of the list : ",len(lst))

**Output:**

Enter number of elements : 3

The elements are :

45

23

87

List : [45, 23, 87]

The elements in the list are :

45

23

87

Enter the element to insert : 95

Enter the index to insert at : 1

After insertion : [45, 95, 23, 87]

Enter the element to remove : 23

After removing : [45, 95, 87]

Enter the element to search : 87

Element is present at index : 2

Enter the element to remove : 45

Enter the element to insert : 13

Enter the index to insert at : 0

After replacing : [13, 95, 87]

The size of the list : 3

**Conclusion:**

A linked list is a linear data structure consisting of nodes where each node contains a data element and a reference (link) to the next node in the sequence. Linked lists offer dynamic memory allocation and efficient insertion and deletion operations compared to arrays.Linked lists can grow or shrink in size dynamically as elements are added or removed. Insertion and deletion operations are generally faster in linked lists compared to arrays, especially for large lists. Linked lists only allocate memory for the data elements and the references, avoiding the need for contiguous memory allocation.