| 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 |





             





**Code:**

class Node:

def \_\_init\_\_(self, val, next=None) -> None:

self.val = val

self.next = None

class LinkedList:

def \_\_init\_\_(self) -> None:

self.l = []

self.head = None

def append(self, val):

newNode = Node(val)

if not self.l or not self.head:

self.head = newNode

self.l.append(newNode)

else: self.l.append(newNode)

print("Value added successfully")

def remove(self, val):

if not self.l or not self.head:

print("Empty linked list")

return

for i in self.l:

if i.val == val:

self.l.remove(i)

print("value removed successfully")

return

print("value not present")

def travese(self):

if not self.l or not self.head:

print("The linked list is empty")

for i in self.l:

print(f"{i.val}", end=" ")

def search(self, val):

if not self.l or not self.head:

print("Linked list is empty")

for i in range(len(self.l)):

if self.l[i].val == val:

print(f"Value found at Node {i}")

return

print("The value not found")

def replace(self, prev, new):

if not self.l or not self.head:

print("Linked list is empty")

temp = self.head

for i in self.l:

if i.val == prev:

temp = i

break

temp.val = new

print("Value changed successfully")

ll = LinkedList()

flag = True

while flag:

ch = int(input("\nEnter a choice: \n1. Append \n2. Remove \n3. Replace \n4. Traverse \n5. Search \n6. Exit \n"))

if ch == 1:

a = int(input("Enter the value: "))

ll.append(a)

elif ch == 2:

a = int(input("Enter the value to remove: "))

ll.remove(a)

elif ch == 3:

a = int(input("Enter the og value: "))

b = int(input("Enter the new value: "))

ll.replace(a, b)

elif ch == 4:

ll.travese()

elif ch == 5:

a = int(input("Enter the value to search: "))

ll.search(a)

elif ch == 6:

flag = False

else:

print("Invalid input")

**Output:**

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

1

Enter the value: 6

Value added successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

1

Enter the value: 8

Value added successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

1

Enter the value: 5

Value added successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

1

Enter the value: 89

Value added successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

1

Enter the value: 45

Value added successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

4

6 8 5 89 45

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

2

Enter the value to remove: 5

Value removed successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

4

6 8 89 45

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

3

Enter the og value: 8

Enter the new value: 56

Value changed successfully

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

4

6 56 89 45

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

5

Enter the value to search: 89

Value found at Node 2

Enter a choice:

1. Append

2. Remove

3. Replace

4. Traverse

5. Search

6. Exit

6

**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.