PROGRAM: To implement operations of linked list

```
class Node:
  def __init__(self, dataval=None):
    self.dataval = dataval
    self.nextnode = None


class SLinkedList:
  def __init__(self):
    self.headval = None #no head


  def listprint(self):
    printval = self.headval
    while printval is not None:
      print (printval.dataval)
      printval = printval.nextnode


  def AtBegining(self,newdata):
     NewNode = Node(newdata)
     NewNode.nextnode = self.headval
     self.headval = NewNode


  def AtEnd(self, newdata):
     NewNode = Node(newdata)
     if self.headval is None:
        self.headval = NewNode
        return
     lastnode = self.headval
     while(lastnode.nextnode):
        lastnode = lastnode.nextnode
     lastnode.nextnode=NewNode
```

```
def AtGiven(self,newdata,middle_node):
    NewNode = Node(newdata)
    NewNode.nextnode = middle_node.nextnode
    middle_node.nextnode = NewNode

def RemoveHead(self):
    head=self.headval
    self.headval=head.nextnode
    headval=None

def RemoveTail(self):
    node=self.headval
    while(node.nextnode.nextnode != None):
       node=node.nextnode
    lastnode=node.nextnode
    node.nextnode=None

def RemoveAfter(self,prev):
    node=self.headval
    prev_node=node
    while(node is not None):
      if prev_node.dataval !=prev:
          prev_node=node
          node=node.nextnode
      else:
          break
    prev_node.nextnode=node.nextnode

def RemoveGiven(self,key):
    node=self.headval
    prev_node=node
```

```
        while(node is not None):
          if node.dataval !=key:
               prev_node=node
               node=node.nextnode
          else:
               break
        prev_node.nextnode=node.nextnode

   def SearchGiven(self,key):
        node=self.headval
        while(node is not None):
          if node.dataval==key:
               print("Value Present")
               break;
          node=node.nextnode

   def ReplaceGiven(self,source,destination):
        node=self.headval
        while(node is not None):
          if node.dataval==source:
               node.dataval=destination
               break;
          node=node.nextnode

list_Days = SLinkedList()
e1=Node("Mon")
list_Days.headval = e1
e2 = Node("Tue")
e1.nextnode = e2
e3 = Node("Wed")
e2.nextnode = e3
```

```
print("1. Display the list (Traversal):")
list_Days.listprint()

print("2.Insert at beginning: ")
list_Days.AtBegining("Sun")
list_Days.listprint()

print("3.Insert at End: ")
list_Days.AtEnd("Sat")
list_Days.listprint()

print("4.Insert After Wed: ")
list_Days.AtGiven("Thu",e3)
list_Days.listprint()

print("5.Search Thu")
list_Days.SearchGiven("Thu")

print("6.Replace Thu with Sun")
list_Days.ReplaceGiven("Thu","Sun")
list_Days.listprint()


print("7.Delete from beginning ")
list_Days.RemoveHead()
list_Days.listprint()

print("8.Delete from end ")
list_Days.RemoveTail()
list_Days.listprint()
```

```
print("9.Delete after Tue")
list_Days.RemoveAfter("Tue")
list_Days.listprint()

print("10.Delete Tue")
list_Days.RemoveGiven("Tue")
list_Days.listprint()
```

OUTPUT:

```
1. Display the list (Traversal):
Mon
Tue
Wed
2.Insert at beginning:
Sun
Mon
Tue
Wed
3.Insert at End:
Sun
Mon
Tue
Wed
Sat
4.Insert After Wed:
Sun
Mon
Tue
Wed
Thu
```

```
Sat
5.Search Thu
Value Present
6.Replace Thu with Sun
Sun
Mon
Tue
Wed
Sun
Sat
7.Delete from beginning
Mon
Tue
Wed
Sun
Sat
8.Delete from end
Mon
Tue
Wed
Sun
9.Delete after Tue
Mon
Tue
Sun
10.Delete Tue
Mon
Sun
```

PROGRAM: To implement Stack data structure

```
class Stack:
    def __init__(self):
        self.items=[]
    def push(self,item):
        self.items.append(item)
    def pop(self):
        self.items.pop()
    def peek(self):
        top=self.items[-1]
        print(f"The top element is {top}")
    def disp(self):
        print(self.items)
    def search(self,x):
        if x in self.items:
            print(f"Element {x} is present at index {self.items.index(x) -1}")
        else:
            print("Not Present")

stack =Stack()
print('Initial Stack')
stack.disp()

print('After adding few elements')
stack.push("Mon")
stack.push("Tue")
stack.push("Wed")
stack.push("Thu")
stack.disp()

print('Last element popped from stack:')
```

```
stack.pop()
stack.disp()

print('Peek Function:')
stack.peek()

print('Search Sun:')
stack.search("Sun")

print('Search Wed:')
stack.search("Wed")
```

OUTPUT:

```
Initial Stack
[]
After adding few elements
['Mon', 'Tue', 'Wed', 'Thu']
Last element popped from stack:
['Mon', 'Tue', 'Wed']
Peek Function:
The top element is Wed
Search Sun:
Not Present
Search Wed:
Element Wed is present at index 1
```

PROGRAM: To implement Queue data structure

```
class Queue:
   def __init__(self):
      self.items = []

   def display(self):
      print(self.items)

   def enqueue(self, item):
      self.items.append(item)

   def dequeue(self):
      self.items.pop(0)

   def search(self,x):
      if x in self.items:
         print(f"Element {x} is present at index {self.items.index(x)}")
      else:
         print("Not Present")

print("Initial Queue:")
queue = Queue()
queue.display()

print("After adding a few elements")
queue.enqueue("Jan")
queue.enqueue("Feb")
queue.enqueue("Mar")
queue.enqueue("Apr")
queue.display()
```

```
print("After poping the first element")
queue.dequeue()
queue.display()
print("To search an element")
x=input("Which element do you want to search? ")
queue.search(x)
```

OUTPUT:

```
Initial Queue:
[]
After adding a few elements
['Jan', 'Feb', 'Mar', 'Apr']
After poping the first element
['Feb', 'Mar', 'Apr']
To search an element
Which element do you want to search? May
Not Present
```

PROGRAM: To use deque class from collections

```
from collections import deque

d = deque()

print("Initial Queue:")

print(d)

print("After Adding a few elements")

d.extend(["March","April"])

d.extendleft(["Feb","June"])

print(d)


print("Adding element at Front")

d.appendleft("Jan")

print(d)


print("Adding element at Rear")

d.append("May")

print(d)


print("Removing element at Rear")

d.pop()

print(d)


print("Removing element at Front")

d.popleft()

print(d)


x=input("Enter the element you want to search: ")

if x in d:

   print(f"Element {x} is present at index {d.index(x)+1}")

else:

   print("Not Present")
```

OUTPUT:

```
Initial Queue:
deque([])
After Adding a few elements
deque(['June', 'Feb', 'March', 'April'])
Adding element at Front
deque(['Jan', 'June', 'Feb', 'March', 'April'])
Adding element at Rear
deque(['Jan', 'June', 'Feb', 'March', 'April', 'May'])
Removing element at Rear
deque(['Jan', 'June', 'Feb', 'March', 'April'])
Removing element at Front
deque(['June', 'Feb', 'March', 'April'])
Enter the element you want to search: March
Element March is present at index 3
```