```
# Create a node
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None


class LinkedList:

    def __init__(self):
        self.head = None

    # Insert at the beginning
    def insertAtBeginning(self, new_data):
        new_node = Node(new_data)

        new_node.next = self.head
        self.head = new_node

    # Insert after a node
    def insertAfter(self, prev_node, new_data):
        new_node = Node(new_data)

        if prev_node is None:
            print("The given previous node must inLinkedList.")
            return

        if prev_node == 0:
            new_node.next = self.head
            self.head = new_node
            return
        current = self.head
        for _ in range(prev_node - 1):
            if current is None:
                raise ValueError("Invalid position in the linked list.")
            current = current.next
    new_node.next = current.next
    current.next = new_node

# Insert at the end
def insertAtEnd(self, new_data):
    new_node = Node(new_data)

    if self.head is None:
        self.head = new_node
        return

    last = self.head
    while (last.next):
        last = last.next

    last.next = new_node

# Deleting a node
def deleteNode(self, position):

    if self.head is None:
        return

    temp = self.head

    if position == 0:
        self.head = temp.next
        temp = None
        return

    # Find the key to be deleted
    for i in range(position - 1):
        temp = temp.next
        if temp is None:
            break

    # If the key is not present
```

```
        if temp is None:
            return

        if temp.next is None:
            return

        next = temp.next.next

        temp.next = None

        temp.next = next

    # Search an element
    def search(self, key):

        current = self.head

        while current is not None:
            if current.data == key:
                return True

            current = current.next

        return False

    # Sort the linked list
    def sortLinkedList(self, head):
        current = head
        index = Node(None)

        if head is None:
            return
        else:
            while current is not None:
                # index points to the node next to current
                index = current.next
            while index is not None:
                if current.data > index.data:
                    current.data, index.data = index.data, current.data

                index = index.next
            current = current.next

    # Print the linked list
    def printList(self):
        temp = self.head
        while (temp):
            print(str(temp.data) + " ", end="")
            temp = temp.next


if __name__ == '__main__':

    llist = LinkedList()
    llist.insertAtEnd(1)
    llist.insertAtBeginning(2)
    llist.insertAtBeginning(3)
    llist.insertAtEnd(4)
    llist.insertAfter(3, 5)

    print('linked list:')
    llist.printList()

    print("\nAfter deleting an element:")
    llist.deleteNode(3)
    llist.printList()

    print()
    item_to_find = 3
    if llist.search(item_to_find):
        print(str(item_to_find) + " is found")
    else:
        print(str(item_to_find) + " is not found")
```

```
llist.sortLinkedList(llist.head)
print("Sorted List: ")
llist.printList()
```