Program 8:
Implementation of Singly Linked List (SLL) (Traversing the Nodes, searching for a Node, Prepending Nodes, and Removing Nodes)

```
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
class SinglyLinkedList:
    def __init__(self):
        self.first = None
    def insertFirst(self, data):
        temp = Node(data)
        temp.next=self.first
        self.first=temp
    def removeFirst(self):
        if(self.first== None):
            print("list is empty")
        else:
            cur=self.first
            self.first=self.first.next
            print("the deleted item is",cur.data)
    def display(self):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        while(cur):
            print(cur.data, end = " ")
            cur = cur.next
    def search(self,item):
        if(self.first== None):
            print("list is empty")
            return
        cur = self.first
        while cur != None:
            if cur.data == item:
                print("Item is Present in the Linked list")
                return
            else:
                cur = cur.next
        print("Item is not present in the Linked list")
#Singly Linked List
sll = SinglyLinkedList()
while(True):
    ch = int(input("\nEnter your choice 1-insertfirst 2-delete 3-search 4-display 5-exit :"))
    if(ch == 1):
        item = input("Enter the element to insert:")
        sll.insertFirst(item)
        sll.display()
    elif(ch == 2):
        sll.removeFirst()
        sll.display()
    elif(ch == 3):
        item = input("Enter the element to search:")
        sll.search(item)
    elif(ch == 4):
```

```
        sll.display()
    else:
        break
```

Program9:
Implementation of linked list Iterators

```
class Node:
    def __init__(self, data = None):
        self.data = data
        self.next = None
class LinkedList:
    def __init__(self):
        self.first = None
    def insert(self, data):
        temp = Node(data)
        temp.next=self.first
        self.first=temp
    def __iter__(self):
        cur = self.first
        while cur:
            yield cur.data
            cur = cur.next
# Linked List Iterators
ll = LinkedList()
ll.insert(10)
ll.insert(100)
ll.insert(99)
ll.insert("welcome to")
ll.insert("govt polytechnic Bellary")
ll.insert(456.35)
ll.insert(545)
ll.insert(5)
for x in ll:
    print(x)
```