**Exercise 10.2**

1. **Explain** **Explain why the dynamic-set operation INSERT on a singly linked list can be im- plemented in time, but the worst-case time for DELETE is Theta(n)**

**Ans)** Insert in a singly linked list has a fixed number of operations to make the head point to the newly created element and to make the newly created element to point to the next element.

INSERT(L, x):

if L.head == NIL:

L.head = x

x.next = NIL

else:

x.next = L.head.next

L.head = x

Here, we only need to make two assignments therefore constant run time.

When deleting, in the worst case, the element we are deleting might be at the end of the linked list and we may need to traverse the entire list.

DELETE(L,x):

temp = L.head

if temp.key == x.key:

head = head.next

else

while temp.next.key!=x.key:

temp = temp->next

temp->next = temp->next->next

Here, in the worst case, the while loop runs n times.

1. **Implement a stack using singly linked list. The operations push and pop should still take O(1) time. Do you need to add any attributes to the list?**

**Ans)**

Let L be the linked list attribute of the stack

PUSH(S,x):

LIST-PREPEND(S.L , x)

POP(S)

if S.L.head ==NIL:

error “underflow”

else

x = S.L.head

S.L.head = S.L.head.next

return x

No need to add any attributes.

1. **Implement a queue using single linked list. The operations ENQUEUE and DEQUEUE should still take O(1) time. Do you need to add any attributes to the list?**

**Ans)**

// Insert at the end of the linked list and remove from starting

ENQUEUE(Q, x):

if Q.L.head == NIL:

Q.L.head = x

else:

Q.L.tail.next = x

Q.L.tail = x

x.next = NIL

DEQUEUE(Q):

if Q.L.head ==NIL:

error “underflow”

else

x = Q.L.head

if Q.L.head == Q.L.tail:

Q.L.tail = NIL

Q.L.head = Q.L.head.next

return x

**4.** **The dynamic-set operation UNION takes two disjoint sets S1*S*1​ and S2*S*2​ as input, and it returns a set S=S1∪S2 ​ consisting of all the elements of S1*S*1​ and S2*S*2​. The sets S1*S*1​ and S2*S*2​ are usually destroyed by the operation. Show how to support UNION in O(1)*O*(1) time using a suitable list data structure.**

**Ans)**

UNION(S1, S2):

temp = S1.NIL.prev.next

S1.NIL.prev.next = S2.NIL.next

S2.NIL.next.prev = temp

S2.NIL.prev.next = S1.NIL

S1.NIL.prev = S2.NIL.prev

1. **Give a theta(n) time non recursive procedure that reverses a singly linked list of n elements. The procedure should use no more than constant storage beyond that needed for the list itself.**

**Ans.**

REVERSE(L):

if L.head==NIL:

break

else:

curr = L.head

before = NIL

while curr!=NIL:

after = L.next

curr->next = before

before = curr

curr = after

L.head = before