Shahriar Ahmed

ID: 20101588

Section: 4

Theory Assignment - 1

Task-1 → Subtask-(a)

a = 88 , b = 20

↓ last two digit ↓ first two digit

Value	25	103	52	25	0	0	45	25	5	19	93	5	26
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 6 size = 11

Value	25	103	52	25	0	0	0	45	5	19	93	5	26
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 7 size = 10

Value	26	103	52	25	0	0	0	0	45	5	19	93	5
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 8 size = 9

Value	5	26	103	52	0	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 8

Task-1 → Subtask-(b)

Value	5	26	103	52	0	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 0 size = 8

Value	20	5	26	103	52	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 9

Task-1 → Subtask-(c)

Value	20	5	26	103	52	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 9

Value	20	5	26	0	103	52	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 10

Value	20	5	26	103	52	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 9

Task - 1 → Subtask - (d)

Value	20	5	26	103	52	0	0	0	0	45	5	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 9

Value	93	20	5	26	103	52	0	0	0	45	5	48	19
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9, size = 10

Task - 1 → Subtask - (e)

Value	93	20	5	26	103	52	0	0	0	45	5	48	19
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9, size = 10

Value	20	5	26	103	52	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 9

Value	20	26	103	52	0	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 8

Task - 1 → Subtask - (f)

Value	20	26	103	52	0	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 8

Value	20	26	103	0	0	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 7

Task – 1 → Sub task – (g)

Value	20	26	103	0	0	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 9 size = 7

Value	93	20	26	103	0	0	0	0	0	0	45	48	19
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 10 size = 7

Value	19	93	20	26	103	0	0	0	0	0	0	45	48
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 11 size = 7

Value	48	19	93	20	26	103	0	0	0	0	0	0	45
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 12 size = 7

Task-1 → Subtask – (h)

Value	48	19	93	20	26	103	0	0	0	0	0	0	45
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

start = 12 size = 7

Value	19	93	20	26	103	0	0	0	0	0	0	45	48
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

Value	93	20	26	103	0	0	0	0	0	0	45	48	19
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

Value	20	26	103	0	0	0	0	0	0	45	48	19	93
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

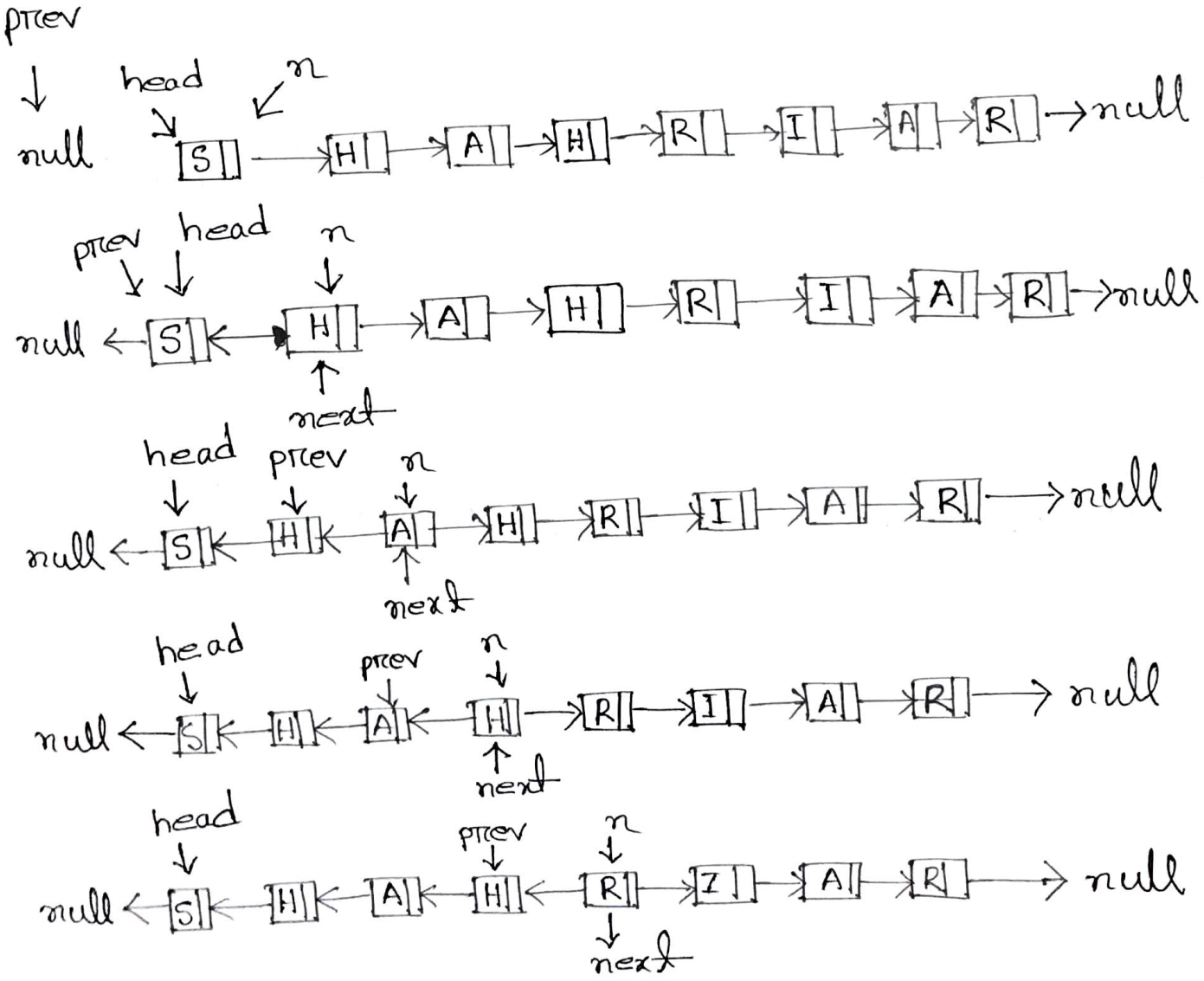
Value	26	103	0	0	0	0	0	0	45	48	19	93	20
Index	0	1	2	3	4	5	6	7	8	9	10	11	12

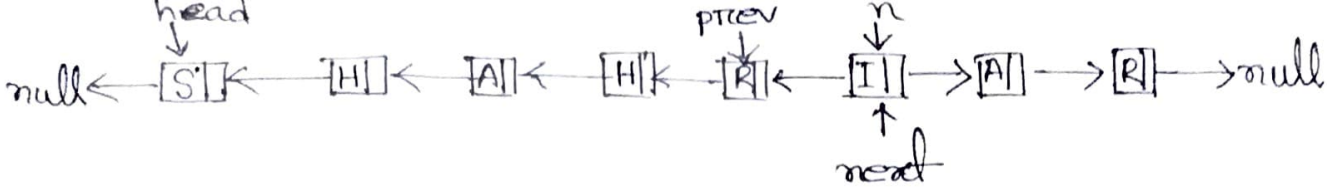
Task-3

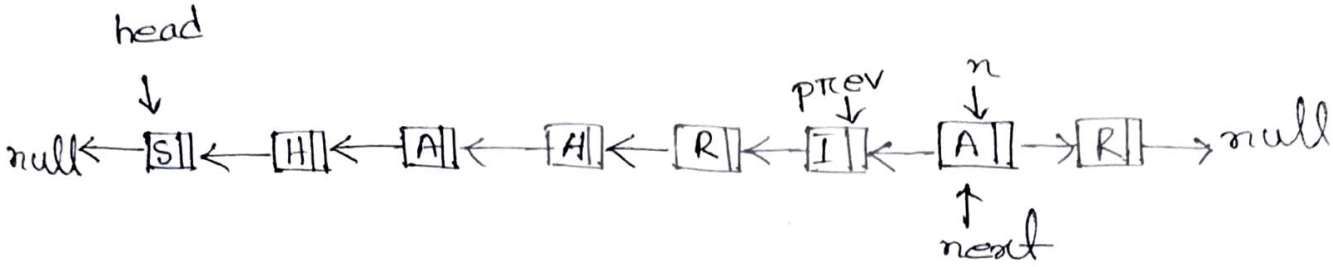
subtask-(a)

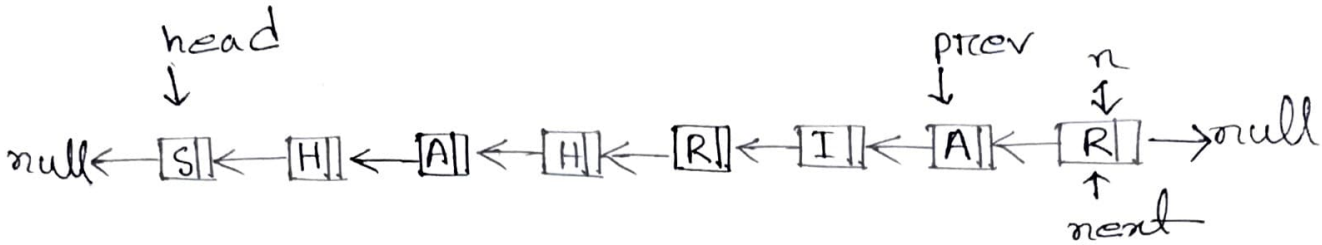
Name : Shahriar

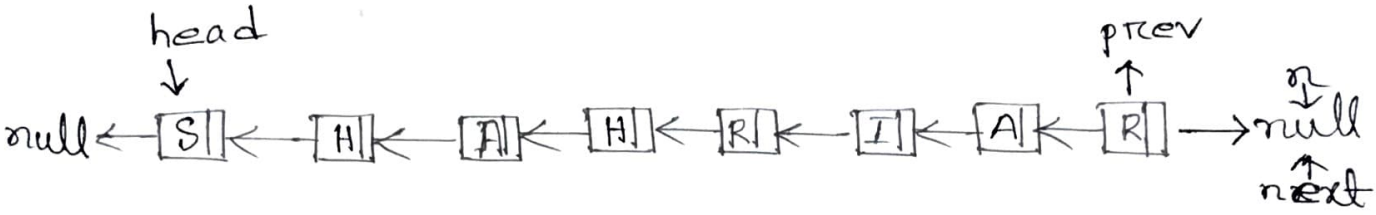

```
n = head
while n is not None:
    next = n.next
    n.next = previous
    previous = n
    n = next
head = previous
```

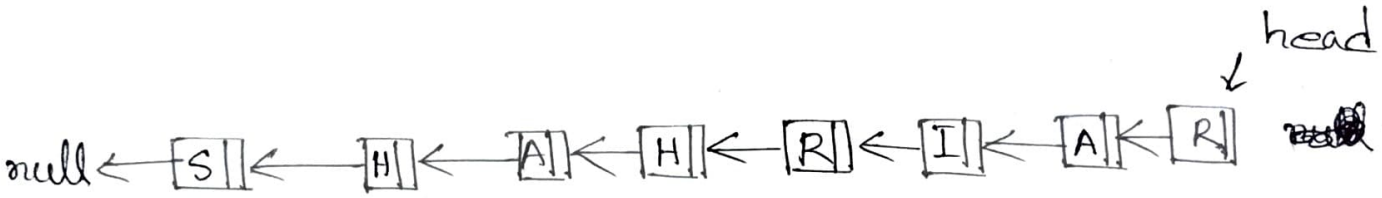






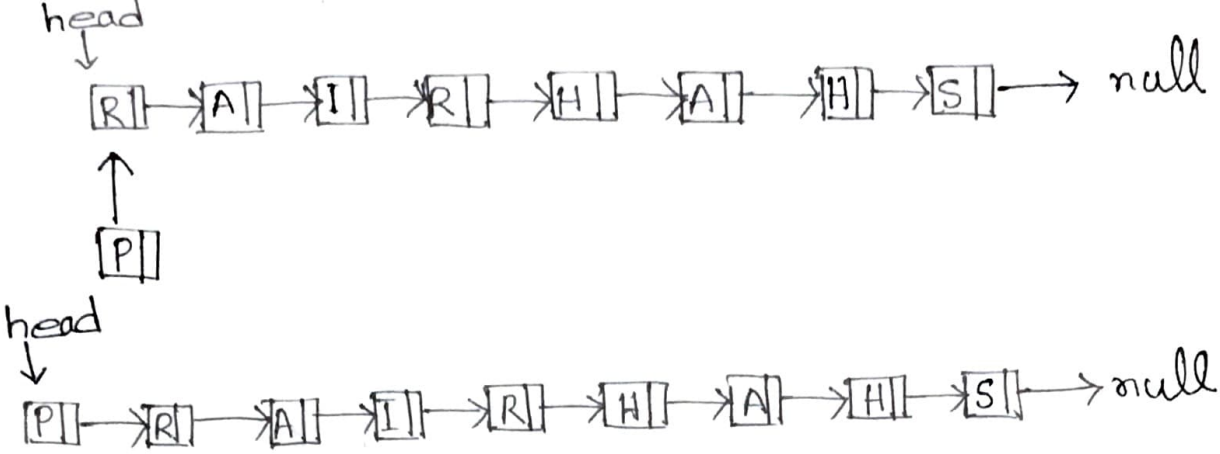




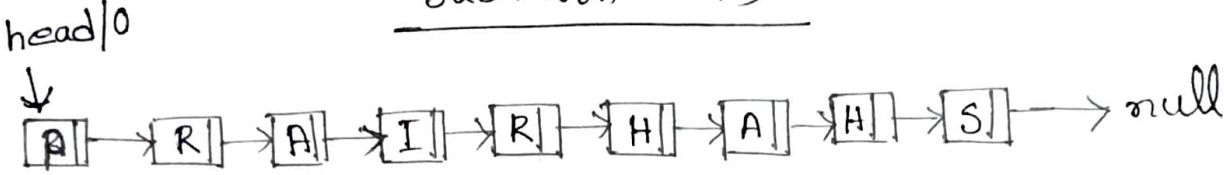


subtask - (b)


```
newNode = Node(i)
if self.head == None:
    self.head = newNode
else:
    n = self.head
    while n.next is not None:
        n = n.next
    n.next = newNode
```

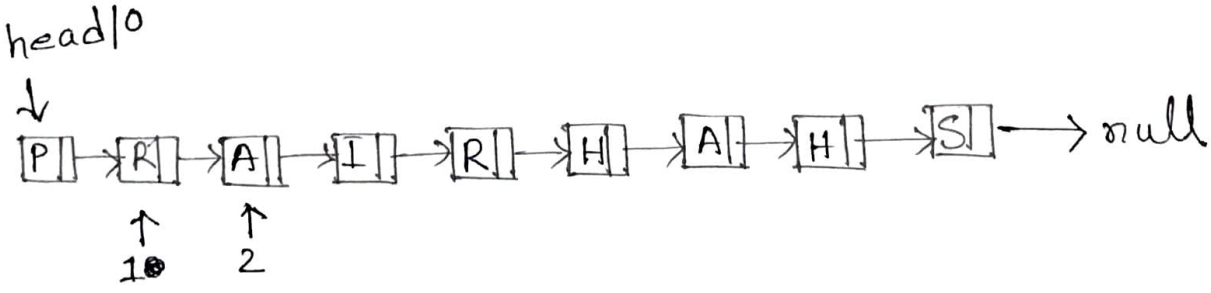


subtask — (c)

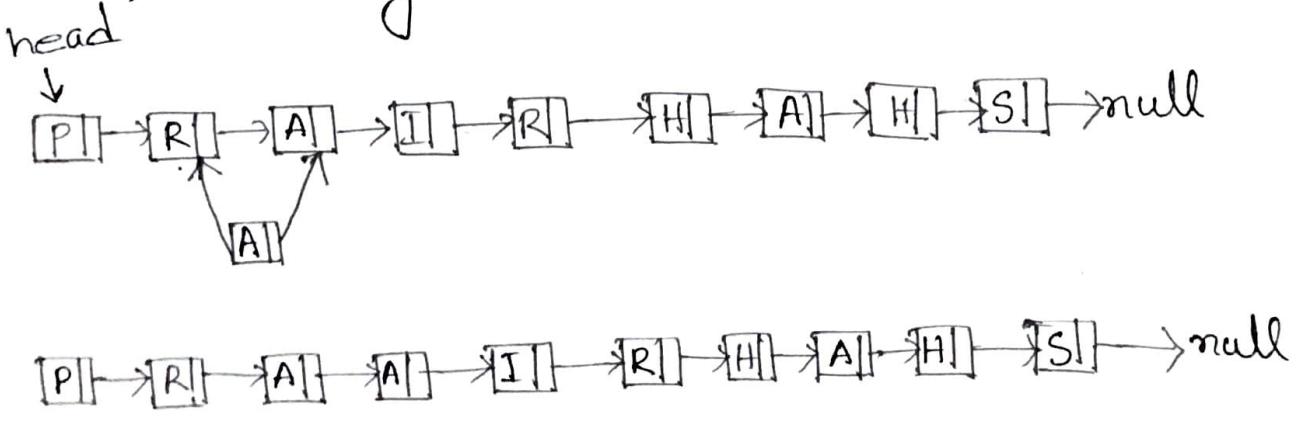


→ traversing linked list

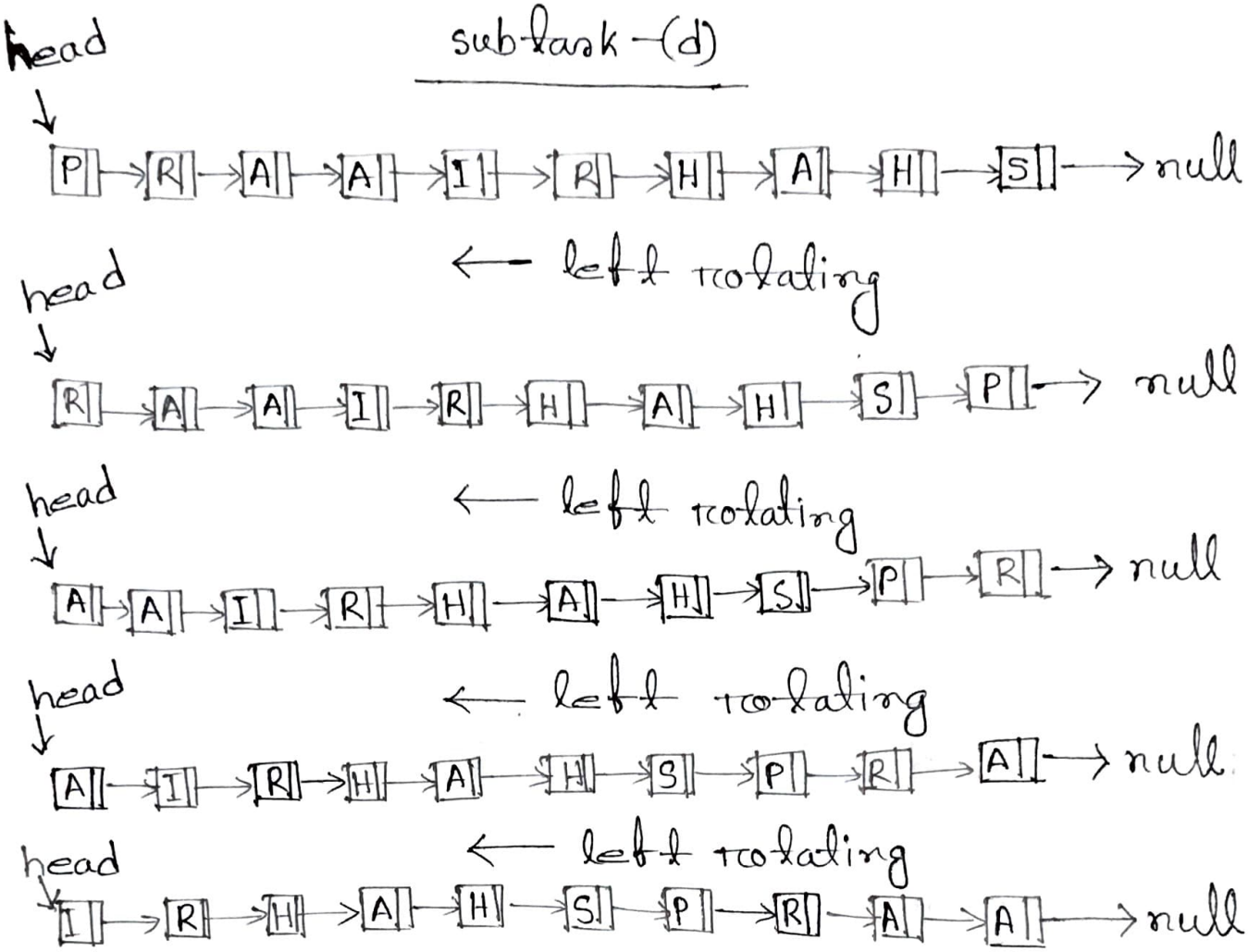
```
newNode = Node(i)
newNode.next = n.next
n.next = newNode
```



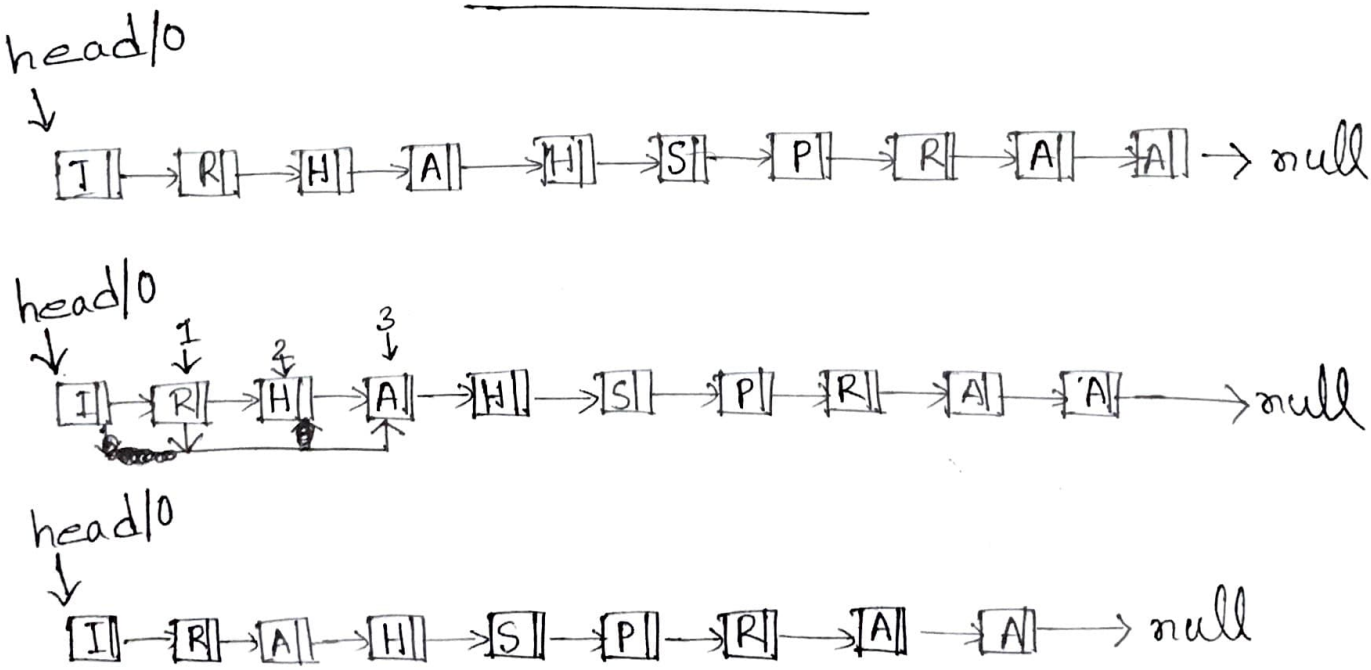
→ traversing linked list



subtask – (d)

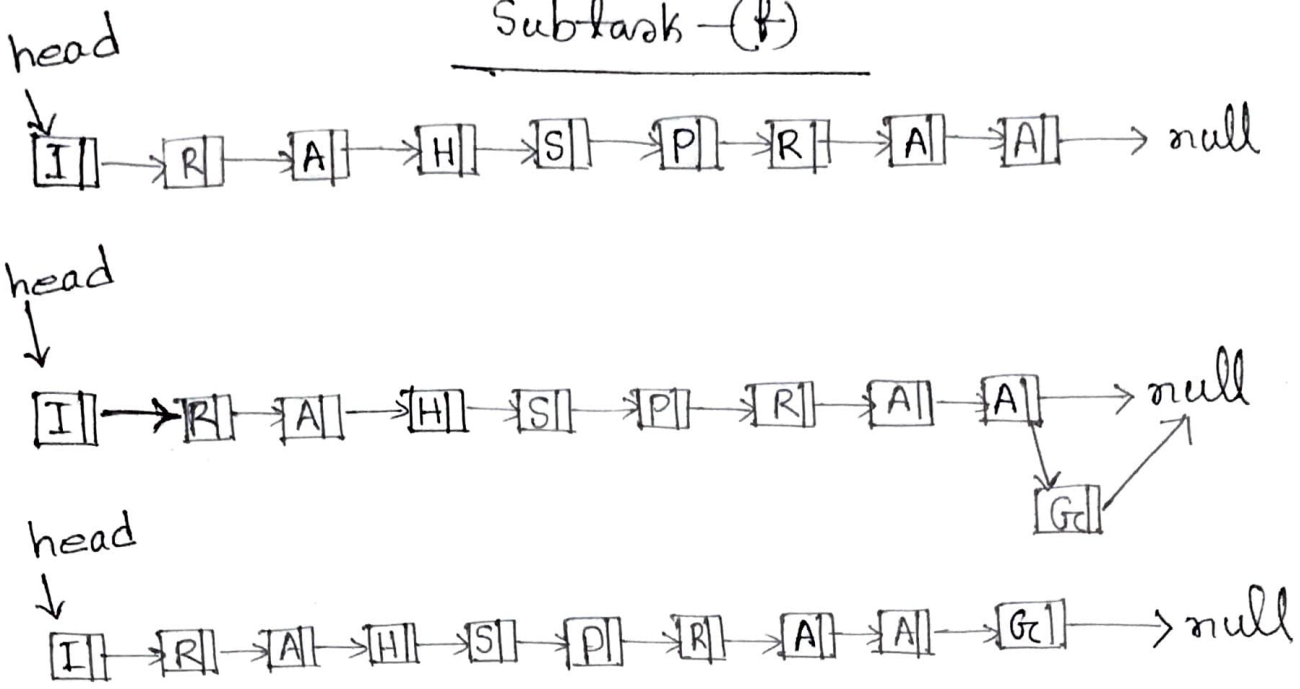


subtask – (e)

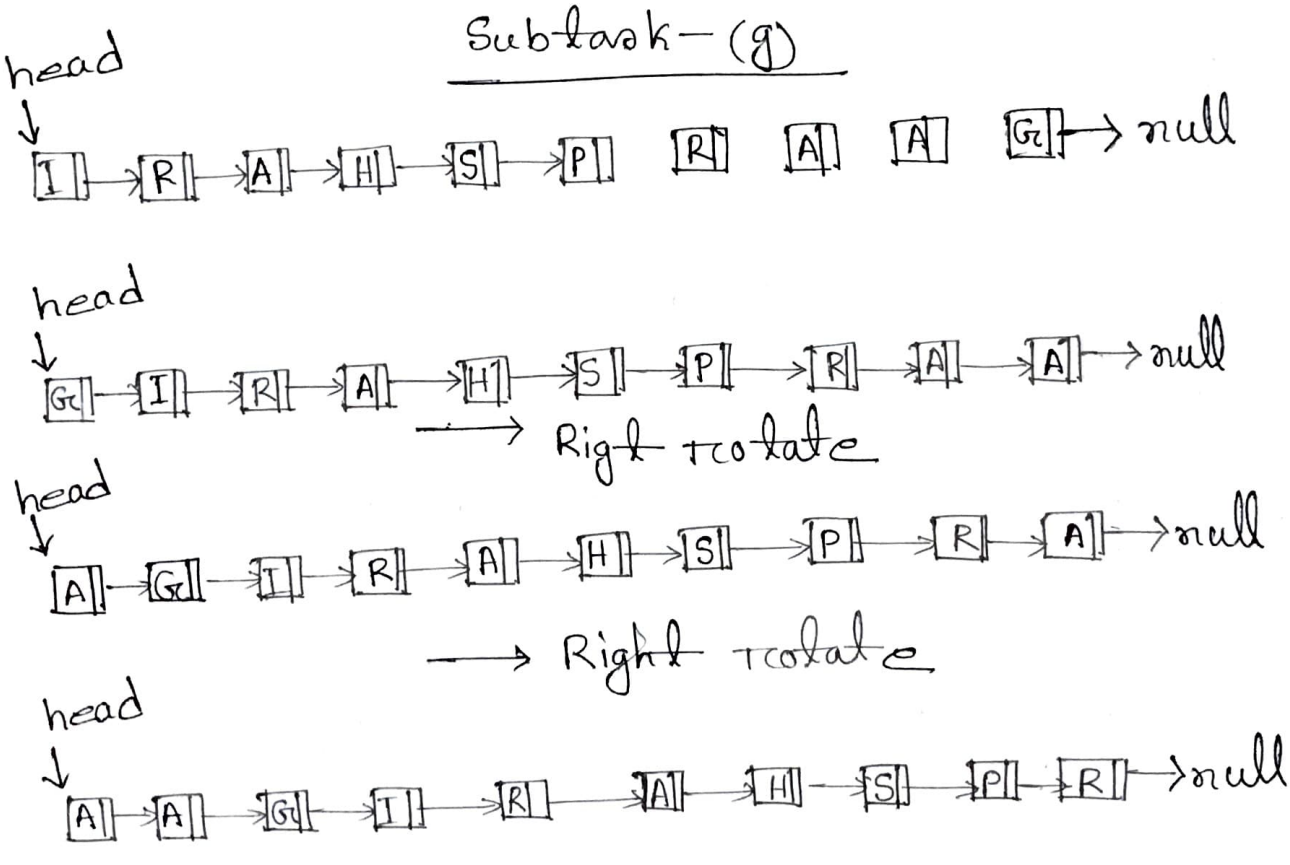


deleting given element

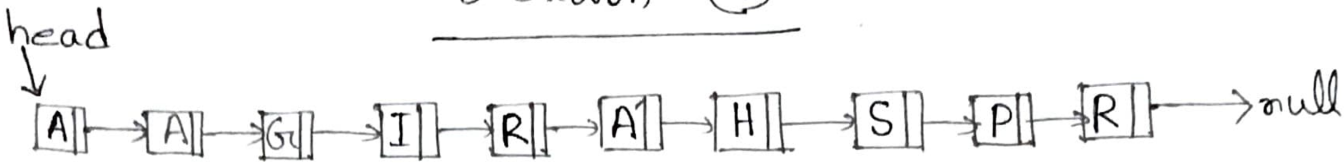
Subtask – (f)



Subtask – (g)



subtask-(b)

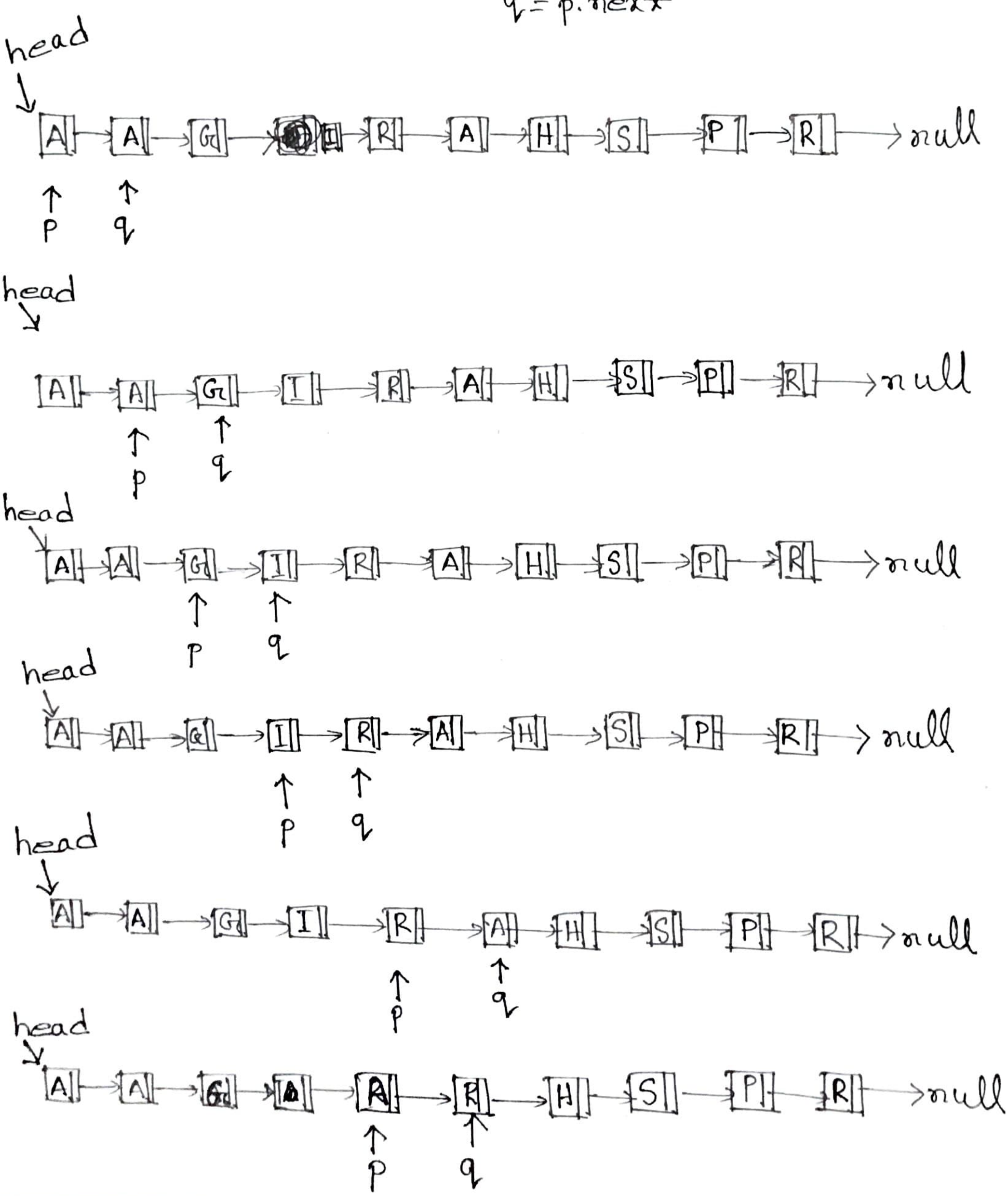


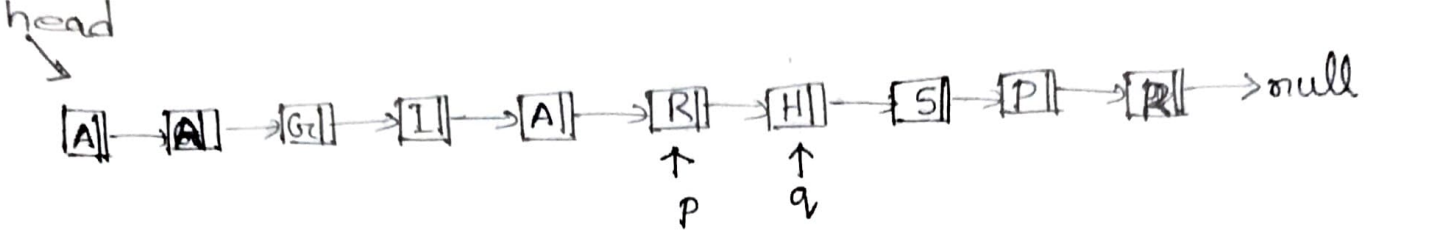
References: p, q, end

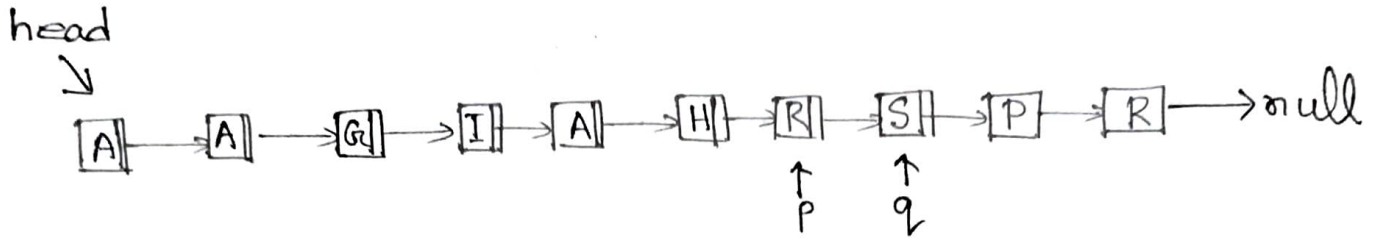
stop when end refers to p.link == end

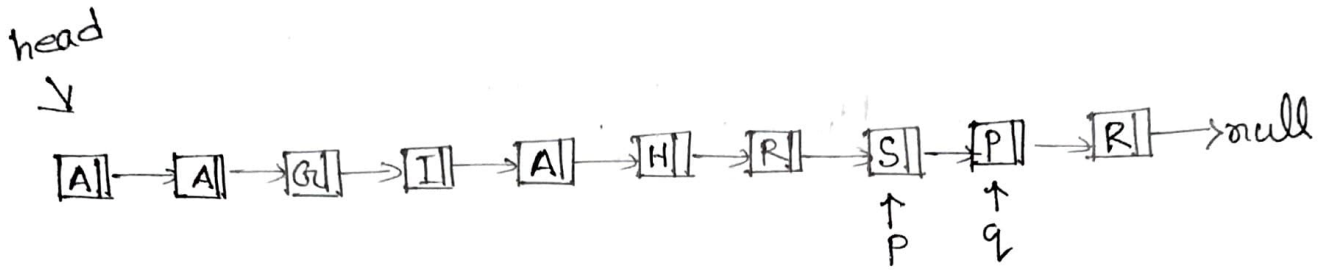
p = self.head

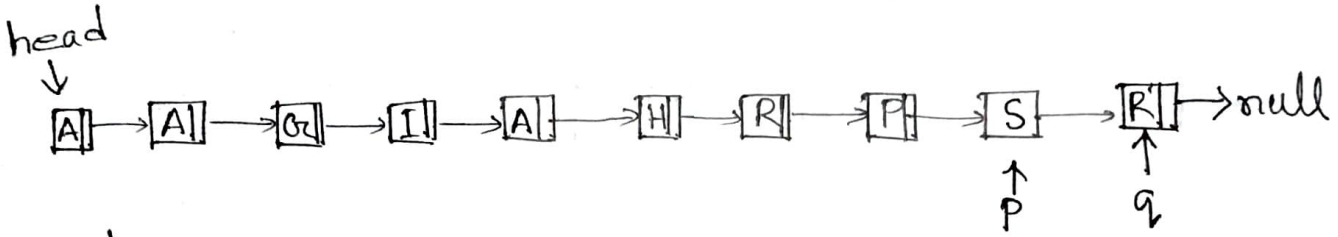
q = p.next

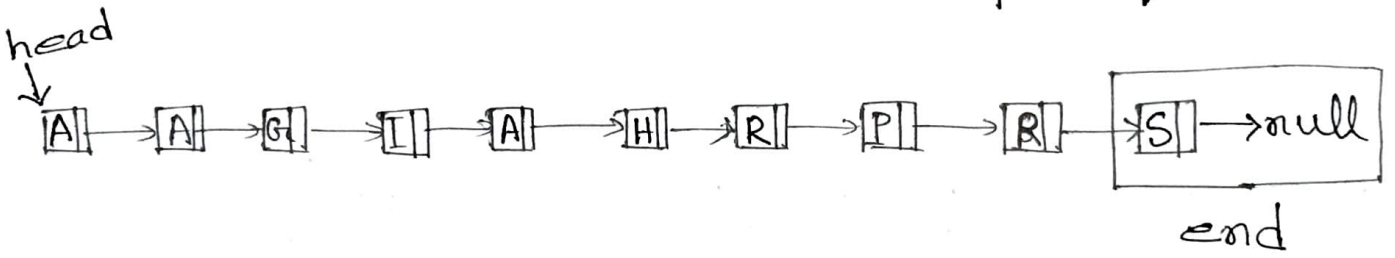


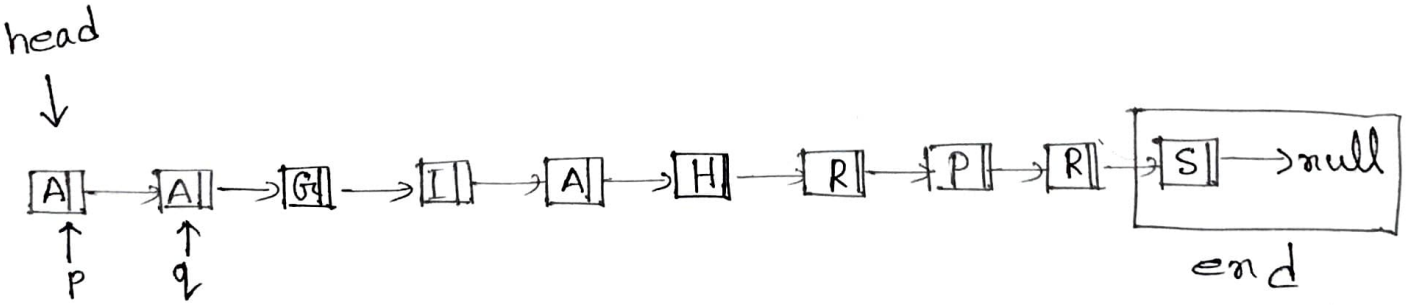

head
A
A
G
I
A
R
H
S
P
R
null
p
q

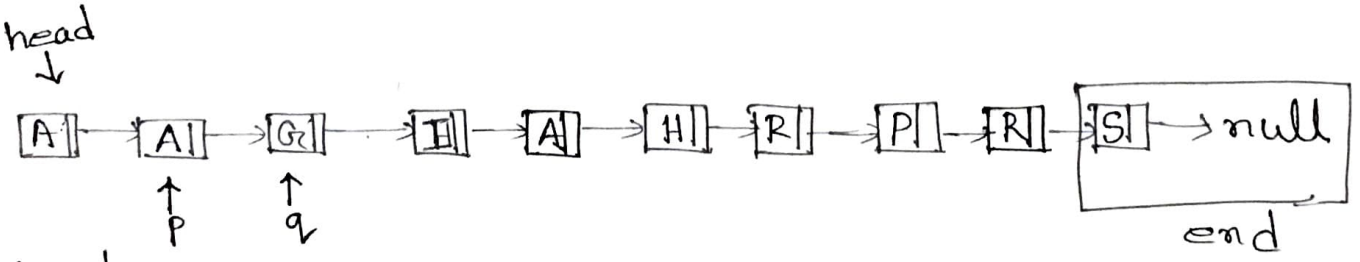

head
A
A
G
I
A
H
R
S
P
R
null
p
q

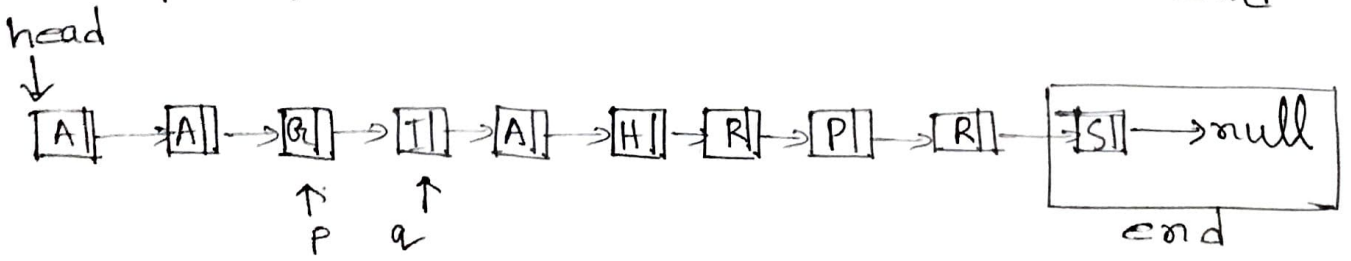

head
A
A
G
I
A
H
R
S
P
R
null
p
q

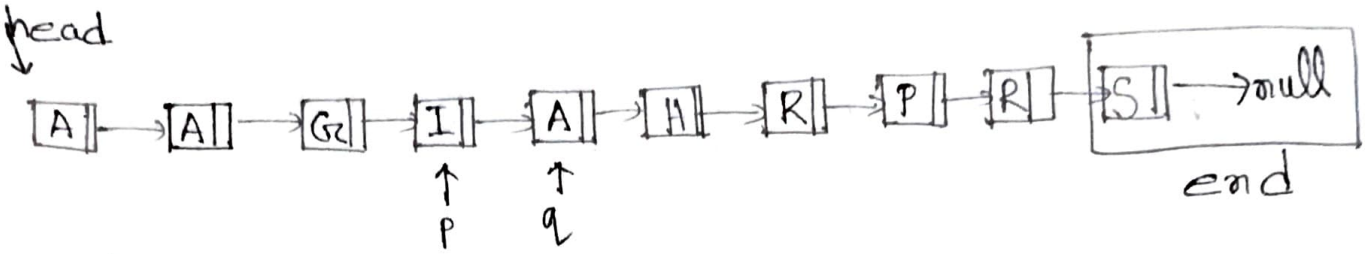

head
A
A
G
I
A
H
R
P
S
R
null
p
q

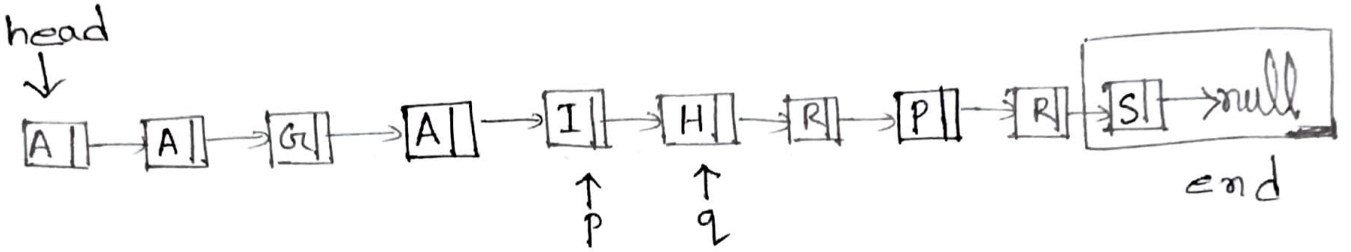

head
A
A
G
I
A
H
R
P
R
S
null
end

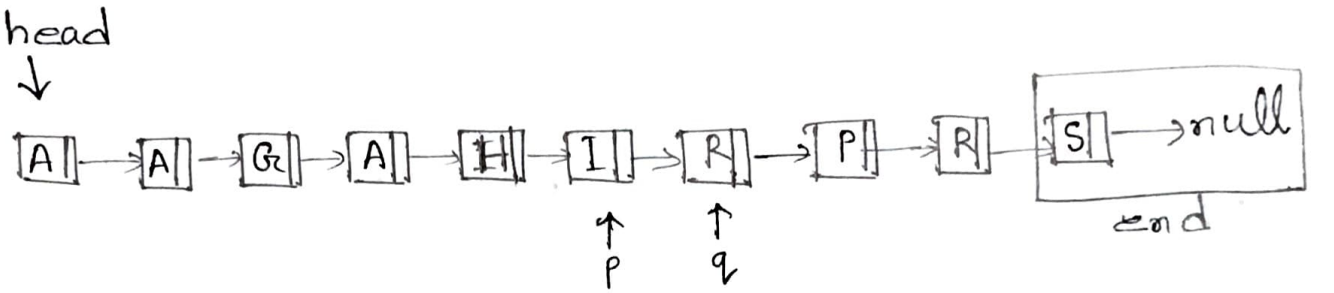

head
A
A
G
I
A
H
R
P
R
S
null
p
q
end

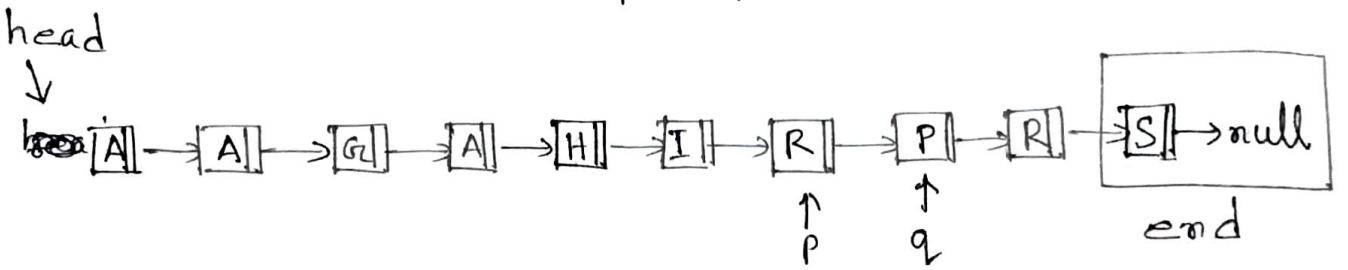

head
A
A
G
I
A
H
R
P
R
S
null
p
q
end

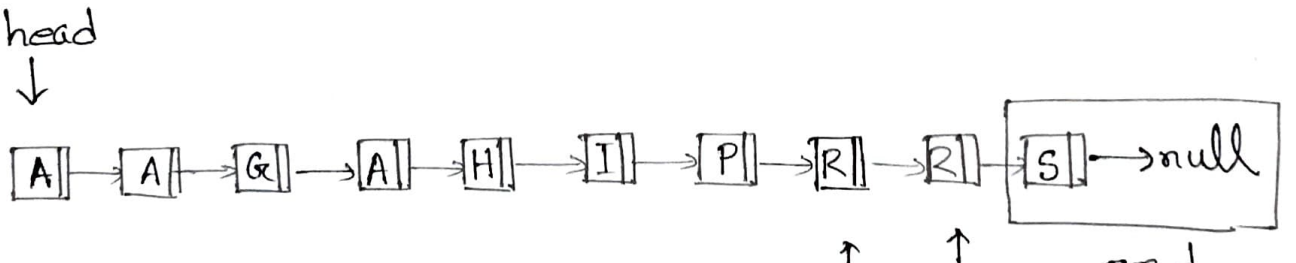

head
A
A
G
I
A
H
R
P
R
S
null
p
q
end

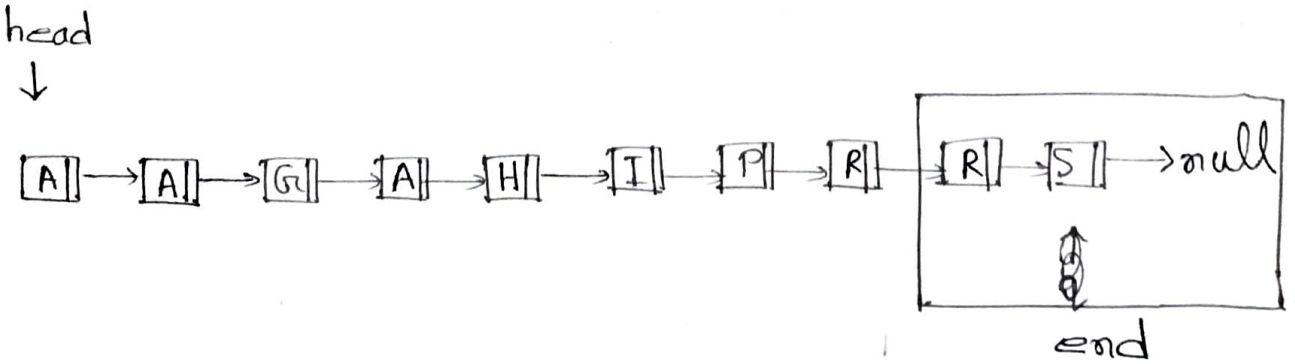

head
A
A
G
I
A
H
R
P
R
S
null
p
q
end

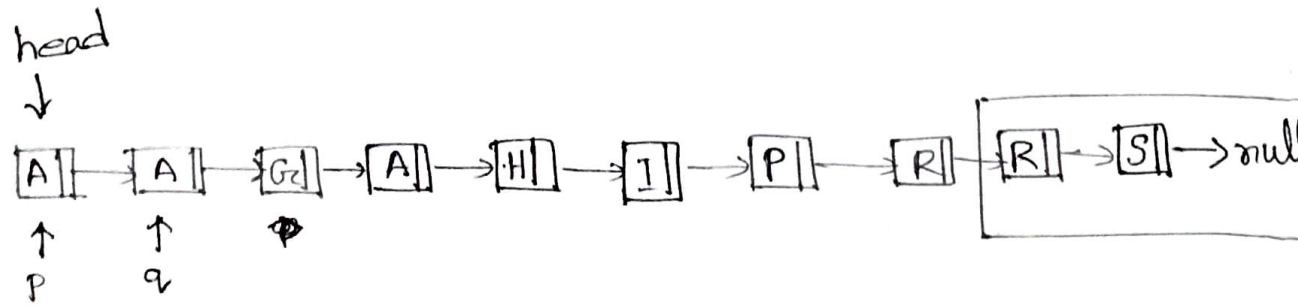

head
A
A
G
A
I
H
R
P
R
S
null
p
q
end

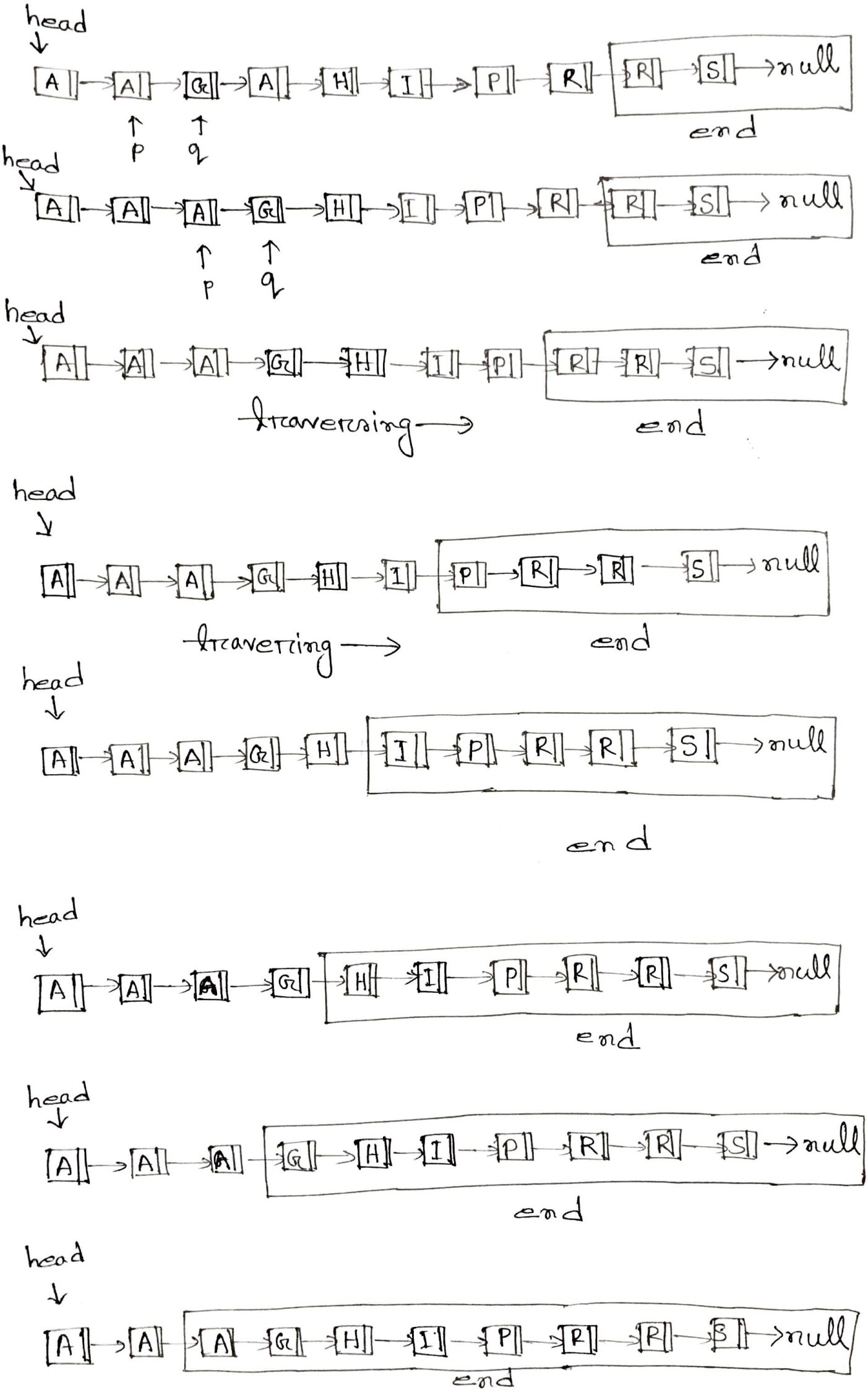

head
A
A
G
A
H
I
R
P
R
S
null
p
q
end

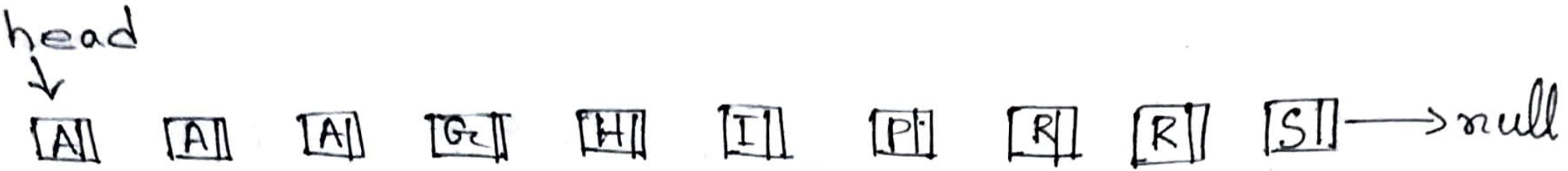

head
A
A
G
A
H
I
R
P
R
S
null
p
q
end


head
A
A
G
A
H
I
P
R
R
S
null
p
q
end


head
A
A
G
A
H
I
P
R
R
S
null
end


head
A
A
G
A
H
I
P
R
R
S
null
p
q


head
A A G A H I P R R S → null
p q
end
head
A A A G H I P R R S → null
p q
end
head
A A A G H I P R R S → null
traversing →
end
head
A A A G H I P R R S → null
traversing →
end
head
A A A G H I P R R S → null
end
head
A A A H I P R R S → null
end
head
A A A G H I P R R S → null
end
head
A A A G H I P R R S → null
end



sorted linked list

####Task-2 Musically Chair Game

```
def musically_chair_game():
    Sum_of_rand = 0
    Random = 3
    Arr = [1, 2, 3, 4, 5, 6, 7]
    while Arr.count(0) <= 6:
        Sum_of_rand += Random
        A_rand = ((3 * Random) + Sum_of_rand) % 4
        Random = A_rand
        if A_rand == 1:
            Arr.pop(int(len(Arr) / 2))
            print(Arr)
        if len(Arr) == 1:
            break
    print("WINNER IS :", Arr[0])

musically_chair_game()
```

####Task-4 Print First Duplicate

```
class Node:
    def __init__(self, e, n=None):
        self.element = e
        self.next = n

class MyList:

    def __init__(self, a):
        self.head = None
        tail = None
        for i in a:
            n = Node(i, None)
            if self.head is None:
                self.head = n
                tail = n
            else:
                tail.next = n
                tail = n
```

```
    def firstDuplicate(self, head):
        n = self.head
        while n.element is not None:
            if n.element == n.next.element:
                print(n.element)
                break
            n.element = n.next.element

list1 = MyList([5,4,15,2,3,4])
list1.firstDuplicate([5,4,15,2,3,4])
list2 = MyList([6,6,10,10,1,1,10,6])
list2.firstDuplicate([6,6,10,10,1,1,10,6])
```

####Task-5 Remove Multiple of Five

```
class Node:
    def __init__(self, e, n=None):
        self.element = e
        self.next = n

class MyList:

    def __init__(self, a):
        self.head = None
        tail = None
        for i in a:
            n = Node(i, None)
            if self.head is None:
                self.head = n
                tail = n
            else:
                tail.next = n
                tail = n

    def remove_multiple_of_five(self, head):
        if self.head.element % 5 == 0:
            self.head = self.head.next
        n = self.head
        x = None
```

```
        n = self.head
        while n is not None:
            while n is not None and n.element % 5 != 0:
                x = n
                n = n.next
            x.next = n.next
            n = x.next

        n = self.head
        while n is not None:
            print(n.element)
            n = n.next

list1 = MyList([5,6,35,10,12,90])
list1.remove_multiple_of_five([5,6,35,10,12,90])
```

####Task-6 Sum of Two Dummy Headed Linked List

```
class Node:
    def __init__(self, e, n=None):
        self.element = e
        self.next = n

class MyList:

    def __init__(self, a, b):
        self.head = Node(0)
        self.start = Node(0)
        tail = Node(None)
        for i in a:
            n = Node(i)
            if self.head.next is None:
                self.head.next = n
                tail = n
            else:
                tail.next = n
                tail = n

        for i in b:
            x = Node(i)
```

```
            if self.start.next is None:
                self.start.next = x
                tail = x
            else:
                tail.next = x
                tail = x

        n = self.head.next
        while n is not None:
            print(n.element)
            n = n.next

    def sum(self):
        a = ""
        b = ""
        c = 0

        n = self.head.next
        while n is not None:
            a = a + str(n.element)
            n = n.next

        x = self.start.next
        while x is not None:
            b = b + str(x.element)
            x = x.next

        print(a,b)
        c = int(a) + int(b)
        print(c)
```

####Task-7 Insertion at the Beginning of the Circular Linked List

```
class Node:
    def __init__(self, e, n=None):
        self.element = e
        self.next = n

class MyList:
```

```
    def __init__(self, a):
        self.tail = None

    def showCircularLinkedList(self):
        if self.tail == None:
            print("The Linked List is Empty")
            return

        n = self.tail.next
        while n is not None:
            print(n.element, end=" ")
            n = n.next
            if n == self.tail.next:
                break
        print()

    def insertion_Beginning_Circular(self, element):
        n = Node(element)
        n.next = self.tail.next
        self.tail.next = n
```

####Task-8 Insert Before in DHCDLL

```
class Node:
    def __init__(self, e, n=None, p=None):
        self.element = e
        self.next = n
        self.previous = p

class MyList:
    def __init__(self, a):
        self.head = Node(None)
        self.head.next = self.head
        self.head.previous = self.head

        x = self.head
        for i in a:
            newNode = Node(i)
            newNode.previous = x
            x.next = newNode
```

```
        self.head.previous = x
        x.next = self.head

    def insertBefore(self, elem, newElement):
        n = self.head.next
        x = self.head.next
        y = self.head
        while n is not y:
            if int(n.element) == int(elem):
                insertNode = Node(newElement)
                insertNode.previous = x
                x.next = insertNode
                insertNode.next = n
                n.previous = insertNode
            x = n
            n = n.next

        n = self.head.next
        while n is not self.head:
            print(n.element)
            n = n.next

list1=MyList([1,2,3,4])
list1.insertBefore(3,50)
```