Merge 2 Single linked lists :

Assuming both SSL are in Increasing order nOutput:

List Called
A: 1 —> 3 —> 5 —> 7 —> NULL Second Linked List Called
B: 2 —> 4 —> 6 —> NULL

**After Merge: 1 —>3—> 5—> 7—> 2 —> 4 —> 6 —> NULL **
After Merge: 1 —> 2 —> 3 —> 4 —> 5 —> 6 —> 7 —> NULL

```
merge_sorted(head1, head2)
        if (head1 == null)
                    return head2
        if (head2 == null)
                    return head1
        if (head1.data <= head2.data)
           mergedHead = head1
           head1 = head1.next
         else
            mergedHead = head2
            head2 = head2.next
        end if
        mergedTail = mergedHead
        while (head1 != null && head2 != null)
           temp = null
           if (head1.data <= head2.data)
              temp = head1
              head1 = head1.next
            else
              temp = head2
              head2 = head2.next
           endif
           mergedTail.next = temp;
           mergedTail = temp;
        end while
        if (head1 != null)
           mergedTail.next = head1
        else if (head2 != null)
           mergedTail.next = head2
       endif
        return mergedHead
```

----------*

Given a pointer to the head of a singly-linked list, print each value from the reversed list. If the given list is empty, do not print anything.

Linked List : 1 —> 2—> 3 —> NULL

Example Output Reversed Linked list : 3 —> 2—> 1

```
PrintReversed(head)
#input is head node in linkedlist   output print list reversed
    If(head =null)
        return
    else
        PrintReversed(head.next)
        Print(head.data)
     End if
end
```

----------*

Write an algorithm the deletes nodes with odd values Output:
if the given Linked list is : 10 -> 7 -> 40 -> 9
the list should be : 10-> 40*

```
DeleteODD()
        p=head
        if(p.data mod 2==1)
                deleteFirst()
                p=head
        End if
        c=p.next
        WHILE c!= NULL:
             if c.data mod 2==1
                  p.next=c.next
            end if
        c =p
        p=p.next
        end while
```

----------*

*Write an algorithm that prints alternant nodes first from head to tail , second from tail to head Example: if the given list is : 10 -> 40 -> 30 -> 60

1. 2. 3. 4

Output 1 (from head to tail) : 10 , 30
Output 2 (from tail to head) : 40 , 60

```
printAlter()
```

```
c=head
i=1
WHILE c!= NULL:
      if i mod 2==1
          print (c.data)
      endif
      c=c.next
      i=i+1
end while
```

---

10 -> 40 -> 30 -> 60

```
printAlterFromtail(head,size)
if head=null
          return
else
          printAlterFromtail(head.next,size-1)
          if size mod 2==1
                    print(head.data)
end if
```

60 40

Write an algorithm that adds an input after all odd position Example:

1. 2. 3. 4

if the given list is : 10 -> 40 -> 30 -> 60 if input : 2

Output: 10->2->40->30->2-> 60

Note: Position of original nodes is : 0 -> 1 -> 2 -> 3 then add before position 1 and 3

```
inserAfterOdd(e)
c=head
 i=1
WHILE c!= NULL:
      if i mod 2==1
           n=new Node(e,c.next)
           c.next=n
     end if
    c=c.next
     i=i+1
end while
```