Linked List

Question 1:
Assume the structure of a Linked List node is as follows.

```
struct node {
    int data;
    node * next;
}
```

Suppose that we have a linked lists as shown in the figure:

![Linked List Diagram](image)

Draw the linked list in which case:

a) Insert a node (value of data: 9) at the beginning of linked list.
b) Insert a node (value of data: 10) at the end of linked list.
c) Insert a node (value of data: 15) at the pTemp.
d) Delete the node which have value of data 3.
e) Delete the node which have value of data 17.
f) Delete the node which pTemp pointed.

What is the output of the following code?

g) void fun1(node* head) {
    if (head == NULL)  
        return;
    fun1(head->next);
    printf("%d  ", head->data);
}

h) void fun2(node* head) {
    if (head== NULL) 
        return;
    printf("%d  ", head->data);
    if(head->next != NULL )
        fun2(head->next->next);
    printf("%d  ", head->data);
}

Question 2:
Imagine we have the method to delete an element from a list as shown in the figure. Which of the following code can be used in searching the target element (i.e. pred and tmp pointers)? Explain.

a. 1. pred=null; tmp = head;
   2. loop (tmp is not null and tmp->data is not the target)
      1. pred = pred->next;
      2. tmp = tmp->next;
   3. end loop
b. 1. pred=null; tmp = head;
   2. loop (tmp is not null and tmp->data is not the target)
      1. pred = tmp;
      2. tmp = tmp->next;
   3. end loop

Question 3:
Write a member function which takes a list sorted in increasing order and deletes any duplicate nodes from the list.
Example: Input: 1->1->5->7->7->9->10->NULL
Output: 1->5->7->9->10->NULL

Question 4:
Write a member function to remove all the nodes which have a greater value on right side.
Example: Input: 12->15->10->11->5->6->2->3->NULL
Output: 15->11->6->3->NULL

Question 5:
Write a member function that takes two lists and insert nodes of second list into first list at alternate positions of first list.
Note: Use of extra space is not allowed (Not allowed to create additional nodes).
Example:
<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1: 1-&gt;2-&gt;5-&gt;7-&gt;NULL L2: 4-&gt;10-&gt;NULL</td>
<td>L1: 1-&gt;4-&gt;2-&gt;10-&gt;5-&gt;7-&gt;NULL L2: NULL</td>
</tr>
</tbody>
</table>