Linked List Test 2

Consider the following data fields and methods.

private ListNode front;

public void processList(Integer val)  
 {  
 ListNode temp, prev;  
  
 prev = front;  
 temp = front.getNext();  
  
 while (temp != null)  
 {  
 if (temp.getValue().equals(val))  
 {  
 prev.setNext(temp.getNext());  
 }  
 else  
 {

prev = temp;  
 }  
 temp = temp.getNext();  
 }

}

1. Which of the following best describes what processList does?
2. It removes all consecutive nodes at the front of the list with value val.
3. It removes the first occurrence of the node with value val.
4. It removes all nodes with value val.
5. It removes all nodes with value val except the first such node.
6. Consider modifying processList by adding the following code directly below the ListNode variable declarations:

while (front != null && front.getValue().equals(val))  
 {  
 front = front.getNext();  
 }

if (front == null)  
 return;

Now which of the following best describes what processList does?

1. It removes all consecutive nodes at the front of the list with value val.
2. It removes the first occurrence of the node with value val.
3. It removes all nodes with value val.
4. It removes all nodes with value val except the first such node.
5. Consider the following method:

public void mystery()  
 {  
 ListNode cur = front;  
 ListNode prev = null;

while (cur != null)  
 {  
 ListNode temp = null;  
 if(cur == front)  
 {  
 temp = front.getNext();  
 front.setNext(temp.getNext());  
 temp.setNext(front);  
 front = temp;  
 prev = cur;

}  
 else  
 {  
 temp = cur.getNext();  
 cur.setNext(temp.getNext());  
 temp.setNext(cur);  
 prev.setNext(temp);  
 prev = cur;

}   
 cur=cur.getNext();

}

}

If head refers to the first node of a linked list with five nodes, A → B → C → D → E→ F,   
 which of the following lists is returned by mystery(head)?

1. A → A→ B → B→ C → C
2. B → A → D → C → F → E
3. A → B → C → D → E → F
4. F → E → D → C → B → A
5. Consider the following method.

public void mystery(ListNode node)  
{

if (node != null)  
 {

mystery(node.getNext());

System.out.print(node.getValue() + " ");

}

}

1. Prints the list in its original order
2. Prints the list in reverse order
3. Prints null
4. Does not print anything do to a run-time error
5. Consider the following class:

public class MyLinkedList  
{  
 private ListNode front;  
  
 public MyLinkedList()

{

front = null;  
 }

public void addFirst(Object val)  
 {  
 if (front == null)  
 {

front = new ListNode(val, null);

}

else

{

*< missing code >*

}

}

}

The firstNode method should add new values to the beginning of the linked List. Which of the following   
code segments can replace *< missing code >?*

1. ListNode newNode = new ListNode(val, front);  
    front = newNode;
2. ListNode newNode = new ListNode(val, null);  
    newNode.setNext(front);  
    front = newNode;
3. front = new ListNode(val, front);
4. A, B, and C will all add a new value to beginning of list