Linked List

Use the following Node definition for all problems:

```
struct Node
{
  typedef int Item;
  Item data;
  Node *link;
}
```

1. Write a function to concatenate two linked lists. Given lists l1 = (2, 3, 1) and l2 = (4, 5), after return from concatenate(l1,l2) the list l1 should be changed to be l1 = (2, 3, 1, 4, 5). Your function should not change l2 and should not directly link nodes from l1 to l2 (i.e. the nodes inserted into l1 should be copies of the nodes from l2.)

2.
```
void concatenate(Node*& h1, Node* h2 );
//
// Precondition: h1 and h2 are head pointers of linked lists.
// The lists may be empty or non-empty.
//
// Postcondition: A copy of list h2 is concatenated (added to the end)
// of list h1.  List h2 should be unchanged by the function.
// NOTE: The nodes added to the list h1 must be copies of the
// nodes in list h2.
```

3. Write a function to insert a number into a sorted linked list. Assume the list is sorted from smallest to largest value. After insertion, the list should still be sorted. Given the list l1 = (3, 17, 18, 27) and the value 20, on return l1 be the list (3, 17, 18, 20, 27).

4.
```
void insertInOrder(Node*& head_ptr, int value);
//
// Precondition: head_ptr is the head pointer of a linked list
// sorted in non-decreasing order. The list may be empty or non-empty.
//
// Postcondition: The number value is inserted in the list.
// The list should be sorted on return from the function.
```

5. Write a function to return the median value in a sorted linked list. If the length i of the list is odd, then the median is the ceiling(i/2) member. For example, given the list (1, 2, 2, 5, 7, 9, 11) as input, your function should return the value 5. If the length of the list is even, then the median is the mean of the i/2 and (i/2)+1 members. Thus, the median of the sorted list (2, 4, 8, 9) is (4+8)/2. Finally, define the median of an empty list to be 0.

6.
```
FILL IN RETURN TYPE listMedian ---- FILL IN ARGUMENTS ----------
//
// Precondition: ----  FILL IN -------
//
// Postcondition: ----  FILL IN -------
//
//
```

7. Write a function to reverse the nodes in a linked list. Your function should have time complexity O(n), where n is the length of the list. You should create no new nodes.

8.
```
void reverse(Node*& head_ptr);
//
// Precondition: head_ptr is the head pointer of a linked list.
// The list may be empty or non-empty.
//
```

// **Postcondition:** head_ptr points to the list of Nodes in reverse
// order.