Finding middle element in a linked list

Given a singly linked list of **N** nodes. The task is to find the middle of the linked list. For example, if given linked list is 1->2->3->4->5 then the output should be 3.
If there are even nodes, then there would be two middle nodes, we need to print the second middle element. For example, if given linked list is 1->2->3->4->5->6 then the output should be 4.

Example 1:

```
Input:
LinkedList: 1->2->3->4->5
Output: 3
```

Example 2:

```
Input:
LinkedList: 2->4->6->7->5->1
Output: 7
```

Your Task:
The task is to complete the function **getMiddle**() which takes a head reference as the only argument and should return the data at the middle node of the linked list.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 5000

Reverse a linked list

Given a linked list of **N** nodes. The task is to reverse this list.

Example 1:

```
Input:
N = 6
value[] = {1,2,3,4,5,6}
Output: 6 5 4 3 2 1
Explanation: After reversing the list,
elements are 6->5->4->3->2->1.
```

Example 2:

```
Input:
N = 5
value[] = {2,7,8,9,10}
Output: 10 9 8 7 2
Explanation: After reversing the list,
elements are 10->9->8->7->2.
```

Your Task:
The task is to complete the function **reverseList**() with head reference as the only argument and should return new head after reversing the list.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:
1 <= N <= 10^4

Rotate a Linked List

Given a singly linked list of size **N**. The task is to rotate the linked list counter-clockwise by **k** nodes, where k is a given positive integer smaller than or equal to length of the linked list.

Example 1:

```
Input:
N = 8
value[] = {1,2,3,4,5,6,7,8}
k = 4
Output: 5 6 7 8 1 2 3 4
Explanation:
```

Example 2:

```
Input:
N = 5
value[] = {2,4,7,8,9}
k = 3
Output: 8 9 2 4 7
Explanation:
```

Your Task:
The task is to complete the function **rotate**() which takes a **head reference** as the **first argument and k** as the **second argument**. The **printing** is done **automatically** by the **driver code**.

Expected Time Complexity: O(N).
Expected Auxiliary Space: O(1).

Constraints:

$1 <= N <= 10^3$

$1 <= k <= 10^3$

Reverse a Linked List in groups of given size.

Given a linked list of size **N**. The task is to reverse every **k** nodes (where k is an input to the function) in the linked list.

Example 1:

```
Input:
LinkedList: 1->2->2->4->5->6->7->8
K = 4
Output: 4 2 2 1 8 7 6 5
```

Example 2:

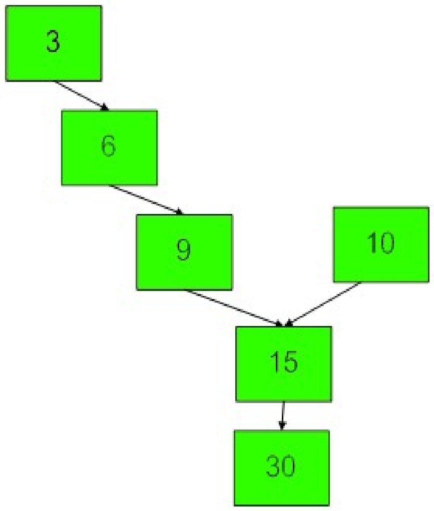
```
Input:
LinkedList: 1->2->3->4->5
K = 3
Output: 3 2 1 5 4
```

Your Task:
The task is to complete the function **reverse**() which should reverse the linked list in group of size **k** and return the head of the modified linked list.

Expected Time Complexity : O(n)
Expected Auxilliary Space : O(1)

Intersection Point in Y Shapped Linked Lists

Given two singly linked lists of size **N** and **M,** write a program to get the point where two linked lists intersect each other.



Above diagram shows an example with two linked list having 15 as intersection point.

Example 1:

```
Input:
LinkList1 = {10,20,5,10}
LinkList2 = {30,40,50,5,10}
Output: 5
Explanation:The point of intersection of
two linked list is 5, means both of them
get linked (intersects) with each other
at node whose value is 5.
```

Your Task:
The task is to complete the function **intersetPoint**() which finds the point of intersection of two linked list. The function should return data value of a node where two linked lists merge. If linked list do not merge at any point, then it should return **-1**.

Challenge : Try to solve the problem without using any extra space.

Expected Time Complexity: O(N+M)
Expected Auxiliary Space: O(1)

Constraints:

1 <= N <= 100

1 <= value <= 1000
