```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x),
next(nullptr) {}
 *     ListNode(int x, ListNode *next) :
val(x), next(next) {}
 * };
 */
class Solution {
    int len(ListNode* head){
        int len=0;
        ListNode* temp=head;
        while(temp){
            temp=temp->next;
            len++;
        }
        return len;
    }
public:
    ListNode* rotateRight(ListNode* head, int
k) {
        if(!head || !head->next) return head;
     //move to len-kth node
       int length=len(head);
       k=k%length;
       //edge case
       if(k==0) return head;//no revere
       int cnt=length-k;
       ListNode* temp=head;
       while(temp){
       cnt--;
       if(cnt==0) break;
       temp=temp->next;
       }
        ListNode* newhead=temp->next;

        temp->next=nullptr;
         ListNode* temp2=newhead;

        while(temp2->next){
            temp2=temp2->next;
        }//now temp is tail so connect to
head
        temp2->next=head;
        return newhead;
    }
};
```

```
class Solution {
public:
    bool hasCycle(ListNode *head) {
        ListNode* fast = head;
        ListNode* slow = head;

        while (fast != nullptr && fast->next
!= nullptr) {
            fast = fast->next->next;
            slow = slow->next;

            if (fast == slow) {
                return true;
            }
        }

        return false;
    }
};
```

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x),
next(nullptr) {}
 *     ListNode(int x, ListNode *next) :
val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteMiddle(ListNode* head) {
      if(head==NULL || head->next==NULL){
        return NULL;
      }
      ListNode*slow=head;
      ListNode*fast=head;
      ListNode*prev=NULL;

      while(fast!=NULL && fast->next!=NULL){
          prev=slow;
          slow=slow->next;
          fast=fast->next->next;
      }
      prev->next=slow->next;
      delete slow;
      return head;
    }

};
```

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode() : val(0), next(nullptr) {}
 *     ListNode(int x) : val(x),
next(nullptr) {}
 *     ListNode(int x, ListNode *next) :
val(x), next(next) {}
 * };
 */
class Solution {
public:
    ListNode* deleteDuplicates(ListNode*
head) {
        ListNode* current = head;
        while(current && current-> next){
            if(current->val == current->next-
>val){
                ListNode* temp = current-
>next; //storing the duplicate value for
deletion.
                current->next = current-
>next->next;
                delete temp; //removing the
repeated val.
            }
            else
            current = current->next;
        }
        return head;

    }
};
```