Rearrange a linked list such that all even and odd positioned nodes are together

Rearrange a linked list in such a way that all odd position nodes are together and all even positions node are together,

Examples:

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

Input:  10->22->30->43->56->70
Output: 10->30->56->22->43->70
```

We strongly recommend that you click here and practice it, before moving on to the solution.

The important thing in this question is to make sure that all below cases are handled
1) Empty linked list
2) A linked list with only one node
3) A linked list with only two nodes
4) A linked list with odd number of nodes
5) A linked list with even number of nodes

The below program maintains two pointers 'odd' and 'even' for current nodes at odd an even positions respectively. We also store first node of even linked list so that we can attach the even list at the end of odd list after all odd and even nodes are connected together in two different lists.

```
// C++ program to rearrange a linked list in such a
// way that all odd positioned node are stored before
// all even positioned nodes
#include<bits/stdc++.h>
using namespace std;

// Linked List Node
struct Node
{
   int data;
   struct Node* next;
};

// A utility function to create a new node
Node* newNode(int key)
{
   Node *temp = new Node;
   temp->data = key;
   temp->next = NULL;
   return temp;
}

// Rearranges given linked list such that all even
// positioned nodes are before odd positioned.
// Returns new head of linked List.
Node *rearrangeEvenOdd(Node *head)
{
   // Corner case
   if (head == NULL)
      return NULL;

   // Initialize first nodes of even and
   // odd lists
   Node *odd = head;
   Node *even = head->next;

   // Remember the first node of even list so
```

```
    // Remember the first node of even list so
    // that we can connect the even list at the
    // end of odd list.
    Node *evenFirst = even;

    while (1)
    {
        // If there are no more nodes, then connect
        // first node of even list to the last node
        // of odd list
        if (!odd || !even || !(even->next))
        {
            odd->next = evenFirst;
            break;
        }

        // Connecting odd nodes
        odd->next = even->next;
        odd = even->next;

        // If there are NO more even nodes after
        // current odd.
        if (odd->next == NULL)
        {
            even->next = NULL;
            odd->next = evenFirst;
            break;
        }

        // Connecting even nodes
        even->next = odd->next;
        even = odd->next;
    }

    return head;
}

// A utility function to print a linked list
void printlist(Node * node)
{
    while (node != NULL)
    {
        cout << node->data << "->";
        node = node->next;
    }
    cout << "NULL" << endl;
}

// Driver code
int main(void)
{
    Node *head = newNode(1);
    head->next = newNode(2);
    head->next->next = newNode(3);
    head->next->next->next = newNode(4);
    head->next->next->next->next = newNode(5);

    cout << "Given Linked List\n";
    printlist(head);

    head = rearrangeEvenOdd(head);

    cout << "\nModified Linked List\n";
    printlist(head);

    return 0;
}
```

Output:

Given Linked List
1->2->3->4->5->NULL
Modified Linked List
1->3->5->2->4->NULL