**Reverse alternate nodes in Link List: -**

**Medium** Accuracy: **48.48%** Submissions: **36K+** Points: **4**

Given a linked list, you have to perform the following task:

1. Extract the alternative nodes starting from second node.
2. Reverse the extracted list.
3. Append the extracted list at the end of the original list.

**Note**: Try to solve the problem without using any extra memory.

**Example 1:**

**Input:**

LinkedList = 10->4->9->1->3->5->9->4

**Output:**10 9 3 9 4 5 1 4

**Explanation:**Alternative nodes in the given linked list are 4,1,5,4. Reversing the alternative nodes from the given list, and then appending them to the end of the list results in a list 10->9->3->9->4->5->1->4.

**Example 2:**

**Input:**

LinkedList = 1->2->3->4->5

**Output:**1 3 5 4 2

**Explanation:**Alternative nodes in the given linked list are 2 and 4. Reversing the alternative nodes from the given list, and then appending them to the end of the list results in a list 1->3->5->4->2.

**Your Task:**  
You don't have to read input or print anything. Your task is to complete the function **rearrange**() which takes the head of the linked list as input and rearranges the list as required.

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

**Constraints:**  
1 <= N <= 105  
0 <= Node\_value <= 109

**Code: -**

//{ Driver Code Starts

#include<stdio.h>

#include<stdlib.h>

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node \*next;

Node(int x){

data = x;

next = NULL;

}

};

struct Node \*start = NULL;

/\* Function to print nodes in a given linked list \*/

void printList(struct Node \*node)

{

while (node != NULL)

{

printf("%d ", node->data);

node = node->next;

}

printf("\n");

}

void insert()

{

int n,value;

cin>>n;

struct Node \*temp;

for(int i=0;i<n;i++)

{

cin>>value;

if(i==0)

{

start = new Node(value);

temp = start;

continue;

}

else

{

temp->next = new Node(value);

temp = temp->next;

}

}

}

// } Driver Code Ends

/\*

reverse alternate nodes and append at the end

The input list will have at least one element

Node is defined as

struct Node

{

int data;

struct Node \*next;

Node(int x){

data = x;

next = NULL;

}

};

\*/

class Solution

{

public:

void rearrange(Node \*list)

{

bool flag = false; // EVEN posn

Node \*ptr = list;

Node \*oddhead = NULL, \*odd = NULL;

Node \*evenhead = NULL, \*even = NULL;

Node \*temp = NULL;

while (ptr) {

temp = ptr->next;

if (!flag) {

if (!even)

evenhead = even = ptr;

else {

even->next = ptr;

even = even->next;

even->next = NULL;

}

}

else {

ptr->next = odd;

odd = ptr;

}

ptr = temp;

flag = !flag;

}

oddhead = odd;

even->next = oddhead;

list = evenhead;

}

};

//{ Driver Code Starts.

int main()

{

int t;

cin>>t;

while (t--) {

insert();

Solution ob;

ob.rearrange(start);

printList(start);

}

return 0;

}

// } Driver Code Ends

**T.C: - O(N)**

**S.C: - O(1)**