**Problem Name:** Odd even linked list

**Topics:** Linked list

**Companies:** Amazon, Microsoft, Apple, Facebook, Google, Oracle, Bloomberg, VMware, Capital one, ByteDance, tiktok

**Level:** Medium

**Language:** C++

**Problem Statement**:

Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.

The **first** node is considered **odd**, and the **second** node is **even**, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.

You must solve the problem in O(1) extra space complexity and O(n) time complexity.

**Input Format:**

First line of the input contain integer n (size of list)

Second line contain n space separated integer list values.

Ex:

5

1 2 3 4 5

**Output Format:** Print linked list after grouping add and even.

**Constraints:**

* n == number of nodes in the linked list
* 0 <= n <= 104
* -106 <= Node.val <= 106

**Examples:**

**Input:** head = [1,2,3,4,5]

**Output:** [1,3,5,2,4]

**Solution:**

**Explanation:** The brute-force way of solving this problem would be just to reconstruct a new linked list consisting of all odd nodes at first and then the even nodes of the original list. For getting the odd nodes, we start at head and increment by two each time. Similarly, for even nodes, we start at head -> next and increment each time by two.

We can either do it by forming an array from original list and then iterating over array (requires less conditions) or directly construct from original list.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* oddEvenList(ListNode\* head) {

        vector<int> arr;

        auto dummy = new ListNode(0), iter = head;

        for(; iter; iter = iter -> next)

            arr.push\_back(iter -> val);

        iter = dummy;

        for(int i = 0; i < arr.size(); i += 2)

            iter = iter -> next = new ListNode(arr[i]);

        for(int i = 1; i < arr.size(); i += 2)

            iter = iter -> next = new ListNode(arr[i]);

        return dummy -> next;

    }

int main() {

    ListNode\* a = NULL;

    ListNode\* result =NULL;

    int n, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    result = oddEvenList(a);

    printList(result);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(N)

**Optimized Solution:**

**Explanation:**

* The idea is to split the linked list into 2 : one containing all odd nodes and other containing all even nodes. And finally, attach the even node linked list at the end of the odd node linked list.
* To split the linked list into even nodes & odd nodes, we traverse the linked list and keep connecting the consecutive odd nodes and even nodes (to maintain the order of nodes) and save the pointer to the first even node.
* Finally, we insert all the even nodes at the end of the odd node list.

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

ListNode\* oddEvenList(ListNode\* head) {

    if(!head || !head->next || !head->next->next) return head;

    ListNode \*odd = head;

    ListNode \*even = head->next;

    ListNode \*even\_start = head->next;

    while(odd->next && even->next){

        odd->next = even->next; //Connect all odds

        even->next = odd->next->next;  //Connect all evens

        odd = odd->next;

        even = even->next;

    }

    odd->next = even\_start;   //Place the first even node after the last odd node.

    return head;

}

int main() {

    ListNode\* a = NULL;

    ListNode\* result =NULL;

    int n, temp;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    result = oddEvenList(a);

    printList(result);

    return 0;

}

**Time Complexity**: O(n)

**Space Complexity:** O(1)