Segregate Even Odd in C++

```
#include <iostream>
using namespace std;

class Node {
public:
    int val;
    Node* next;

    Node(int val) {
        this->val = val;
        this->next = nullptr;
    }
};

Node* segregateEvenOdd(Node* head) {
    if (head == nullptr || head->next == nullptr)
return head;

    Node* dummyEven = new Node(-1);
    Node* dummyOdd = new Node(-1);
    Node* evenTail = dummyEven;
    Node* oddTail = dummyOdd;
    Node* curr = head;

    while (curr != nullptr) {
        if (curr->val % 2 != 0) {
            oddTail->next = curr;
            oddTail = oddTail->next;
        } else {
            evenTail->next = curr;
            evenTail = evenTail->next;
        }

        curr = curr->next;
    }

    evenTail->next = dummyOdd->next;
    oddTail->next = nullptr;

    Node* result = dummyEven->next;
    delete dummyEven;
    delete dummyOdd;
    return result;
}

void push(Node*& head, int new_data) {
    Node* new_node = new Node(new_data);
    new_node->next = head;
    head = new_node;
}

void printList(Node* node) {
    while (node != nullptr) {
        cout << node->val << " ";
        node = node->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
```

What This Code Does

1. Builds a linked list: 6 -> 9 -> 10 -> 11
2. Separates **even** and **odd** numbers.
3. Appends odd list **after** the even list.
4. Prints the result: 6 -> 10 -> 9 -> 11

📦 Linked List Construction (push)

push inserts at the head. So insertion order is:

Push Order	Value Inserted	List After Push
1	11	11
2	10	10 → 11
3	9	9 → 10 → 11
4	6	6 → 9 → 10 → 11

🔄 segregateEvenOdd(head) Dry Run

curr->val	Even/Odd	Action	Even List	Odd List
6	Even	Added to even list	6	-
9	Odd	Added to odd list	6	9
10	Even	Added to even list	6 → 10	9
11	Odd	Added to odd list	6 → 10	9 → 11

Then:

- evenTail->next = dummyOdd->next connects 6 → 10 → 9 → 11
- oddTail->next = nullptr ends the list

🖨️ Final Output from printList(head1)

6 10 9 11

📌 Summary

Before Segregation	After Segregation
6 → 9 → 10 → 11	6 → 10 → 9 → 11

```
    push(head, 11);
    push(head, 10);
    push(head, 9);
    push(head, 6);

    Node* head1 = segregateEvenOdd(head);
    printList(head1);

    return 0;
}
```

6 10 9 11