Delete Continuous Nodes of Linked List which Sum up to Zero

Delete Continuous Nodes of Linked List which Sum up to Zero

There is a Singly Linked List with n nodes, delete the continuous nodes which sum up to zero. Print the updated Linked List, if there is no deletion print the original Linked List.

For eg. :

- 6 → -6 → 8 → 4 → -12 → 9 → 8 → -8
  Output : 9

- 20 → 4 → 6 → -10 → 8 → 9 → 10 → -19 → 10 → -18 → 25
  Output : 20 → 25

SOLUTION :

★ Lets initialise a variable sum and add elements one by one.

★ Here in the first example , sum will be 6 then we move to the next node and add -6 to sum and the sum becomes zero. Make 8 as the head node.

★ Now, sum becomes 8 we keep adding and at -12, sum becomes zero. Make 9 as the head node.

★ Repeat the process, sum is 9, add next element 8 and further add -8. Here we observe that the sum becomes 9, which was also in one of the previous cases. This means the numbers in between summed up to zero. We will link the node where the sum was initially 9 to the node where it again becomes 9.

★ This means we will link the node 9 with the node next to -8.

★ Since -8 is the last node , so 9 will be the only node in the updated linked list.

★ To store the value of sum at various levels of the iteration we will use an unordered map

CODE:

- Append Node with value zero at the starting of the linked list.
- Create an unordered_map to store the sum of the node value till current node , with the reference of the current node, during traversal of the Linked List.
- Traverse the given linked list.
    - If there is a Node with value (sum) present in the unordered_map then delete all the nodes from the node corresponding to value (sum) stored in map, to the current node .

    - If there is no Node with value (sum) present in the unordered_map, then store the current sum , with the current node in the map.

```cpp
#include <bits/stdc++.h>
using namespace std;

// Linled List Node
struct Node{
    int data;
    Node* next;

    Node(int x){
        data = x;
        next = NULL;
    }
};


// Function to print Linked List
void PrintList(Node* head){
    while(head){
        cout<<head->data<<" ";
        head = head->next;
    }
    cout<<endl;
}

```

```cpp

// Function to remove continuous nodes that
// sum up to zero
Node* RemoveSumZeros(Node* head)
{
    // Append node with value zero
    Node* first;   first = new Node(0);
    first->next = head;

    // Create unordered_map to store sum and
    // current node
    unordered_map< int, Node* > map1;
    map1[0] = first;

    int sum = 0;

    while(head)
    {
        sum = sum + head->data;

```

```
45          // sum is present in map
46          if( map1.find( sum ) != map1.end() )
47          {
48              Node* prev = map1[sum];
49              Node* temp = prev;
50
51              // Remove in between nodes from map
52              int value = sum;
53              while(temp != head)
54              {
55                  temp = temp->next;
56                  value = value + temp->data;
57                  if(temp != head)
58                      map1.erase(value);
59              }
60              prev->next = head->next;
61          }
62
63          // sum is not present in map
64          else
65              map1 [sum] = head;
66
67          head = head->next;
68      }
69
70      return first->next;
71  }
```

```
72
73  int main()
74  {
75      cout << "Enter no. of nodes in linked list: " ;
76      int n;     cin >> n;
77      int data; cin>>data;
78      Node* head;  head = new Node(data);
79      Node* ptr = head;
80      for(int i = 0; i<n-1; i++)
81      {
82          cin >> data;
83          ptr->next = new Node(data);
84          ptr = ptr->next;
85      }
86
87      PrintList(head);
88
89      Node* new_head = RemoveSumZeros(head);
90
91      PrintList(new_head);
92  }
93
```

Output:

```
Enter no. of nodes in linked list: 11
20 4 6 -10 8 9 10 -19 10 -18 25
Original Linked List
20 4 6 -10 8 9 10 -19 10 -18 25
Updated Linked List
20 25
```