# Addition and Subtraction using Linked Lists

## Meet Jain, 23BCP093

**Theory**

This program represents numbers as linked lists, where each node stores a single digit in reverse order for easier manipulation. The getnum() function converts user input into a linked list, while printlist() displays the stored number. To perform arithmetic operations, the rev() function reverses the list, ensuring that calculations start from the least significant digit. The add() function sums two numbers digit by digit, handling carry, whereas the subtract() function subtracts corresponding digits, managing borrow when needed. After computation, the result is reversed back to its correct order and displayed, ensuring accurate representation of addition and subtraction.

**Code**

#include <bits/stdc++.h>

using namespace std;

struct Node{

    int data;

    Node\* next;

    Node(int x, Node\* ptr){

        data = x;

        next = ptr;

    }

};

Node\* createnode(int x){

    return new Node(x, NULL);

}

Node\* getnum(){

    cout << "Enter number: ";

    int a;

    cin >> a;

    Node\* head = createnode(a%10);

    int temp = a/10;

    while(temp>0){

        Node\* ptr = createnode(temp%10);

        ptr->next = head;

        head = ptr;

        temp = temp/10;

    }

    return head;

}

void printlist(Node \*head){

    Node\* ptr = head;

    while(ptr){

        cout << ptr->data << " ";

        ptr = ptr->next;

    }

    cout << endl;

}

Node\* rev(Node\* head){

    Node\* prev = nullptr;

    Node\* curr = head;

    Node\* front = nullptr;

    while (curr)

    {

        front = curr->next;

        curr->next = prev;

        prev = curr;

        curr = front;

    }

    return prev;

}

Node\* subtract(Node\* headA, Node\* headB) {

    if (!headA || !headB) return headA ? headA : headB;

    Node\* revA = rev(headA);

    Node\* revB = rev(headB);

    Node\* dummyhead = createnode(0);

    Node\* ptr = dummyhead;

    Node\* a = revA;

    Node\* b = revB;

    int borrow = 0;

    while (a) {

        int diff = a->data - borrow;

        if (b) {

            diff -= b->data;

            b = b->next;

        }

        if (diff < 0) {

            diff += 10;

            borrow = 1;

        } else {

            borrow = 0;

        }

        ptr->next = createnode(diff);

        ptr = ptr->next;

        a = a->next;

    }

    Node\* result = rev(dummyhead->next);

    return result;

}

Node\* add(Node\* headA,Node\* headB){

    if(headA == nullptr || headB == nullptr) return headA ? headA : headB;

    Node\* dummyhead = createnode(0);

    Node\* revA = rev(headA);

    Node\* revB = rev(headB);

    Node\* ptr = dummyhead;

    int carry = 0;

    Node\* a = revA;

    Node\* b = revB;

    while (a || b || carry) {

        int sum = carry;

        if (a) {

            sum += a->data;

            a = a->next;

        }

        if (b) {

            sum += b->data;

            b = b->next;

        }

        ptr->next = createnode(sum % 10);

        carry = sum / 10;

        ptr = ptr->next;

    }

    return rev(dummyhead->next);

}

int main(){

    Node\* headA = getnum();

    Node\* headB = getnum();

    printlist(headA);

    printlist(headB);

    cout << "Options: Addition: 1, Subtraction: 2" << endl;

    int x;

    cin >> x;

    if(x==1){

        Node\* adding = add(headA,headB);

        cout << "Addition: ";

        printlist(adding);

    }

    else{

        Node\* subbing = subtract(headA,headB);

        cout << "Subtraction: ";

        printlist(subbing);

    }

    return 0;

}

**Output**

**A computer screen shot of a black screen

Description automatically generated**

**Analysis**

A piece of paper with writing on it

Description automatically generated