NAME: ADITYA JETHANI
ROLL NO: 21BCP166
DIV-3 G-5
DAA ASSIGNMENT 3
AIM:
Performing various algebraic operations on two linked lists.
Theory:
Performing addition, subtraction, multiplication and exponentiation on two linked lists provided by the user.This includes applying mathematical rules to perform these operations on corresponding nodes in the given lists and storing the output in a new linked list. If the lengths of the given lists are not the same then adjust the values of the shorter list with appropriate number of zeroes required.
PROGRAM:
Addition and Subtraction:

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

// create linked list
struct Node
{
    int data;
    Node *next;
};
```

```
// create a new node

Node *createNode(int data)
{
    Node *newNode = new Node();
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

Node *createLinkedList(string str)
{
    Node *head = NULL;
    for (int i = 0; i < str.length(); i++)
    {
        Node *newNode = createNode(int(str[i]) - 48);
        if (head == NULL)
            head = newNode;
        else
        {
            newNode->next = head;
            head = newNode;
        }
    }
    return head;
}

void printReverseLinkedList(Node *head)
{
    Node *temp = head;
    if (temp == NULL)
        return;
    printReverseLinkedList(temp->next);
    cout << temp->data << " ";
}

void makeLinkedListLengthEqual(Node *head1, Node *head2)
{
    int length1 = 0, length2 = 0;
    Node *temp1 = head1;
    Node *temp2 = head2;
    while (temp1->next != NULL)
```

```
    {
        length1++;
        temp1 = temp1->next;
    }
    length1++;
    while (temp2->next != NULL)
    {
        length2++;
        temp2 = temp2->next;
    }
    length2++;
    if (length1 > length2)
    {
        for (int i = 0; i < length1 - length2; i++)
        {
            Node *newNode = createNode(0);
            temp2->next = newNode;
            temp2 = temp2->next;
        }
    }
    else if (length1 < length2)
    {
        for (int i = 0; i < length2 - length1; i++)
        {
            Node *newNode = createNode(0);
            temp1->next = newNode;
            temp1 = temp1->next;
        }
    }
}

Node *addLinkedLists(Node *head, Node *head1, Node *head2, int carry)
{
    if (head1 == NULL && head2 == NULL)
        return NULL;
    int result = head1->data + head2->data + carry;
    head = createNode(result % 10);
    head->next = addLinkedLists(head->next, head1->next, head2->next,
result / 10);
    return head;
}

int findGreater(Node *head1, Node *head2)
```

```
{
    if (head1 == NULL && head2 == NULL)
        return 0;
    int result = 0;
    result = findGreater(head1->next, head2->next);
    if (result != 0)
        return result;
    if (head1->data > head2->data)
        return -1;
    else if (head1->data < head2->data)
        return 1;
    else
        return 0;
}

Node *subtractLinkedLists(Node *head, Node *head1, Node *head2, int carry)
{
    if (head1 == NULL && head2 == NULL)
        return NULL;
    Node *temp1 = head1, *temp2 = head2;
    int greater = findGreater(head1, head2);
    if (greater == -1)
    {
        int result = head1->data - head2->data - carry;
        if (result < 0)
        {
            result += 10;
            carry = 1;
        }
        else
            carry = 0;
        head = createNode(result);
        head->next = subtractLinkedLists(head->next, head1->next,
head2->next, carry);
        return head;
    }
    else
    {
        int result = head2->data - head1->data - carry;
        if (result < 0)
        {
            result += 10;
            carry = 1;
```

```
        }
        else
            carry = 0;
        head = createNode(result);
        head->next = subtractLinkedLists(head->next, head2->next,
head1->next, carry);
        return head;
    }
}


int main()
{
    string s1, s2;
    cin >> s1 >> s2;
    Node *head1 = createLinkedList(s1);
    Node *head2 = createLinkedList(s2);
    makeLinkedListLengthEqual(head1, head2);
    printReverseLinkedList(head1);
    cout << endl;
    printReverseLinkedList(head2);
    cout << endl
         << "Addition is: ";
    Node *head = addLinkedLists(head, head1, head2, 0);
    printReverseLinkedList(head);
    cout << endl
         << "Subtraction is: ";
    head = subtractLinkedLists(head, head1, head2, 0);
    printReverseLinkedList(head);
    cout << endl;

    return 0;
}
```

OUTPUT:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\Users\jetha> cd "c:\Aditya\College Code\DAA lab\P3\" ; if ($?) { g++ LinkedLi
1234
12
1 2 3 4
0 0 1 2
Addition is: 1 2 4 6
Subtraction is: 1 2 2 2
PS C:\Aditya\College Code\DAA lab\P3>
```

Multiplication:

```cpp
// C++ program to Multiply two numbers
// represented as linked lists
#include <bits/stdc++.h>
using namespace std;

struct Node
{
    int data;
    struct Node *next;
};

struct Node *newNode(int data)
{
    struct Node *new_node =
        (struct Node *)malloc(sizeof(struct Node));
    new_node->data = data;
    new_node->next = NULL;
    return new_node;
}

void push(struct Node **head_ref, int new_data)
{

    struct Node *new_node = newNode(new_data);

    new_node->next = (*head_ref);
```

```
    (*head_ref) = new_node;
}

int reverse(struct Node **head_ref)
{
    struct Node *prev = NULL;
    struct Node *current = *head_ref;
    struct Node *next;
    int len = 0;
    while (current != NULL)
    {
        len++;
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    *head_ref = prev;
    return len;
}

struct Node *make_empty_list(int size)
{
    struct Node *head = NULL;
    while (size--)
        push(&head, 0);
    return head;
}

struct Node *multiplyTwoLists(struct Node *first,
                              struct Node *second)
{

    int m = reverse(&first), n = reverse(&second);

    struct Node *result = make_empty_list(m + n + 1);

    struct Node *second_ptr = second,
```

```
            *result_ptr1 = result, *result_ptr2, *first_ptr;

    while (second_ptr)
    {

        int carry = 0;

        result_ptr2 = result_ptr1;

        first_ptr = first;

        while (first_ptr)
        {

            int mul = first_ptr->data * second_ptr->data + carry;

            result_ptr2->data += mul % 10;

            carry = mul / 10 + result_ptr2->data / 10;
            result_ptr2->data = result_ptr2->data % 10;

            first_ptr = first_ptr->next;
            result_ptr2 = result_ptr2->next;
        }

        if (carry > 0)
        {
            result_ptr2->data += carry;
        }

        result_ptr1 = result_ptr1->next;
        second_ptr = second_ptr->next;
    }

    reverse(&result);
    reverse(&first);
    reverse(&second);

    while (result->data == 0)
```

```
    {
        struct Node *temp = result;
        result = result->next;
        free(temp);
    }

    return result;
}

void printList(struct Node *Node)
{
    while (Node != NULL)
    {
        cout << Node->data;
        if (Node->next)
            cout << "->";
        Node = Node->next;
    }
    cout << endl;
}

// Driver program to test above function
int main(void)
{
    struct Node *first = NULL;
    struct Node *second = NULL;

    int n1, n2;
    cout << "Enter the number of elements in first list: ";
    cout << "The elements will be pushed in the list from the end:" cin >> n1;
    for (int i = 0; i < n1; i++)
    {
        int x;
        cin >> x;
        push(&first, x);
    }
    cout << "First List is: ";
    printList(first);
    cout << "Enter the size of second list: ";
```

```
    cout << "The elements will be pushed in the list from the end:" cin >> n2;
    for (int i = 0; i < n2; i++)
    {
        int x;
        cin >> x;
        push(&second, x);
    }
    cout << "Second List is: ";
    printList(second);

    // Multiply the two lists and see result
    struct Node *result = multiplyTwoLists(first, second);
    cout << "Resultant list is: ";
    printList(result);

    return 0;
}
```

OUTPUT:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

PS C:\Users\jetha> cd "c:\Aditya\College Code\DAA lab\P3\" ; if ($?) { g++ multiplylists.cpp
Enter the number of elements in first list:
The elements will be pushed in the list from the end:3
1
2
5
First List is: 5->2->1
Enter the size of second list:
The elements will be pushed in the list from the end:2
1
2
Second List is: 2->1
Resultant list is: 1->0->9->4->1
PS C:\Aditya\College Code\DAA lab\P3>
```