Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

**Academic Year: 2022-2023**

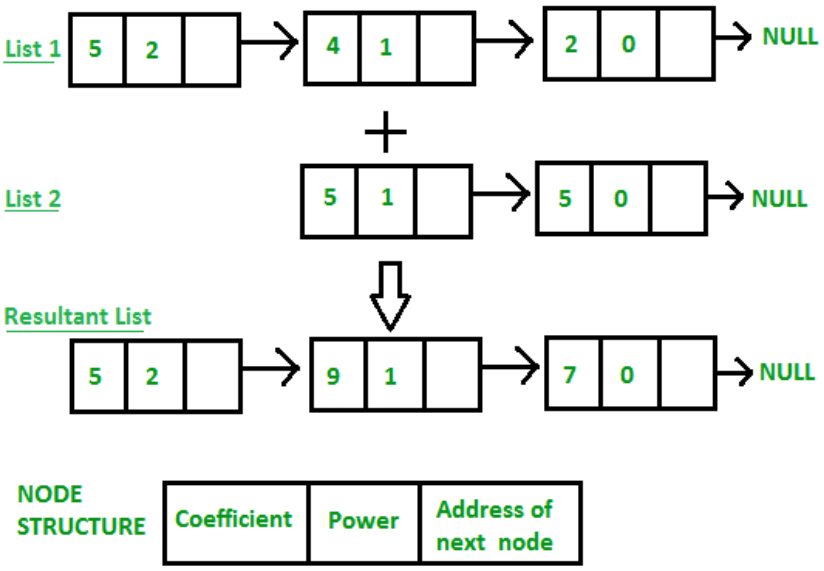
**Name – Prerna Sunil Jadhav** **SAP ID - 60004220127**

**Experiment No – 04**

## AIM: Implementation of Polynomial addition using linked list.

## Theory:

- To add two polynomials that are represented as a linked list, we will add the coefficients of variables with the degree.
- We will traverse both the list and at any step we will compare the degree of current nodes in both the list:
  - ✓ we will add their coefficients if their degree is same and append to the resultant list .
  - ✓ Otherwise, we will append the node with greater node in the resultant list.

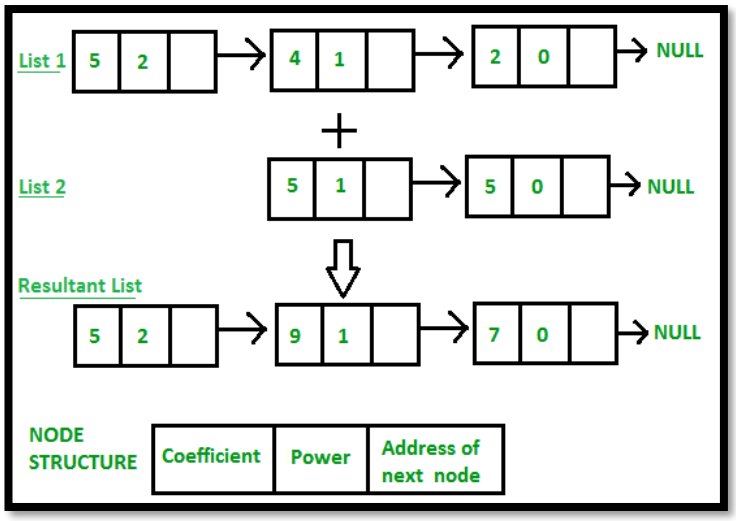



## Algorithm:

1. Create a new linked list, newHead to store the resultant list.
2. Traverse both lists until one of them is null.
3. If any list is null insert the remaining node of another list in the resultant list.
4. Otherwise compare the degree of both nodes, a (first list as *node*) and b (second list as *node*). Here three cases are possible:

- If the degree of a and b is equal, we insert a new node in the resultant list with the coefficient equal to the sum of coefficients of a and b and the same degree.
- If the degree of a is greater than b, we insert a new node in the resultant list with the coefficient and degree equal to that of a.
- If the degree of b is greater than a, we insert a new node in the resultant list with the coefficient and degree equal to that of b.

## Example:




## Program:

```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int coeff;
    int pow;
    struct Node* next;
};

void newnode(int x, int y, struct Node** temp)
{
    struct Node *r, *z;
    z = *temp;
    if (z == NULL) {
        r = (struct Node*)malloc(sizeof(struct Node));
        r->coeff = x;
        r->pow = y;
        *temp = r;
        r->next = (struct Node*)malloc(sizeof(struct Node));
        r = r->next;
        r->next = NULL;
    }
    else {
        r->coeff = x;
        r->pow = y;
        r->next = (struct Node*)malloc(sizeof(struct Node));
        r = r->next;
        r->next = NULL;
    }
```

```
}

void polyadd(struct Node* poly1, struct Node* poly2,
             struct Node* poly)
{
    while (poly1->next && poly2->next) {
        if (poly1->pow > poly2->pow) {
            poly->pow = poly1->pow;
            poly->coeff = poly1->coeff;
            poly1 = poly1->next;
        }

        else if (poly1->pow < poly2->pow) {
            poly->pow = poly2->pow;
            poly->coeff = poly2->coeff;
            poly2 = poly2->next;
        }

        else {
            poly->pow = poly1->pow;
            poly->coeff = poly1->coeff + poly2->coeff;
            poly1 = poly1->next;
            poly2 = poly2->next;
        }

        poly->next
            = (struct Node*)malloc(sizeof(struct Node));
        poly = poly->next;
        poly->next = NULL;
    }
    while (poly1->next || poly2->next) {
        if (poly1->next) {
            poly->pow = poly1->pow;
            poly->coeff = poly1->coeff;
            poly1 = poly1->next;
        }
        if (poly2->next) {
            poly->pow = poly2->pow;
            poly->coeff = poly2->coeff;
            poly2 = poly2->next;
        }
        poly->next
            = (struct Node*)malloc(sizeof(struct Node));
        poly = poly->next;
        poly->next = NULL;
    }
```

```
}

void show(struct Node* node)
{
    while (node->next != NULL) {
        printf("%dx^%d", node->coeff, node->pow);
        node = node->next;
        if (node->coeff >= 0) {
            if (node->next != NULL)
                printf("+");
        }
    }
}

int main()
{
    printf("Prerna Sunil Jadhav - 60004220127\n");
    struct Node *poly1 = NULL, *poly2 = NULL, *poly = NULL;

    newnode(1, 2, &poly1);
    newnode(1, 1, &poly1);
    newnode(1, 0, &poly1);

    newnode(3, 1, &poly2);
    newnode(7, 0, &poly2);

    printf("1st Number: ");
    show(poly1);

    printf("\n2nd Number: ");
    show(poly2);

    poly = (struct Node*)malloc(sizeof(struct Node));

    polyadd(poly1, poly2, poly);

    printf("\nAdded polynomial: ");
    show(poly);

    return 0;
}
```

**OUTPUT:**

```
Prerna Sunil Jadhav - 60004220127
1st Number: 1x^2+1x^1+1x^0
2nd Number: 3x^1+7x^0
Added polynomial: 1x^2+4x^1+8x^0
```

## Conclusion:

A linked list is a data structure that arranges elements in a linear order. In contrast to an array, where the linear order is defined by the array indices, a linked list's linear order is determined by a pointer in each object. When two polynomials are added, the like terms in the two polynomials are combined. We use the term "like terms" to refer to terms that have the same variable and exponent.