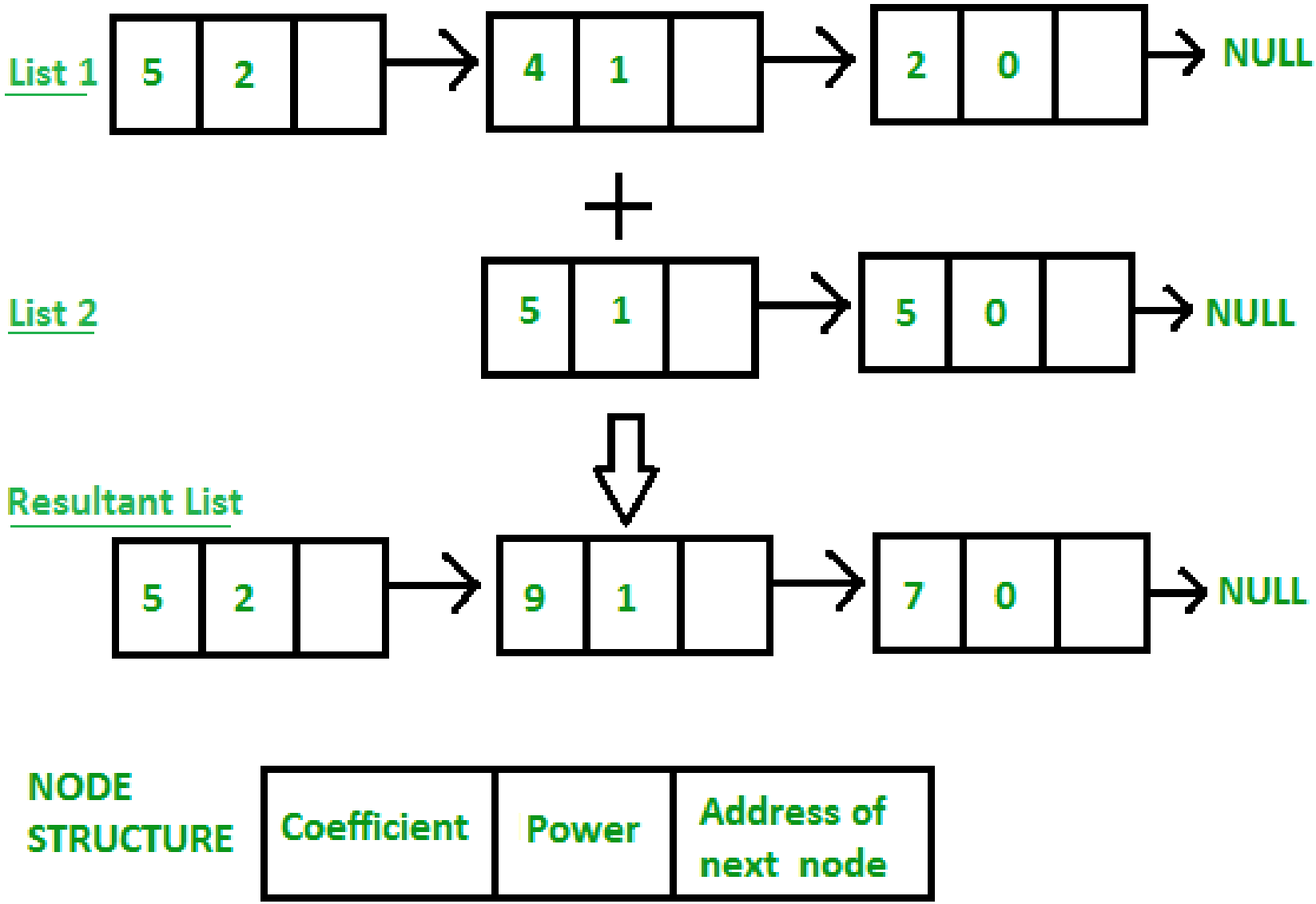
# SLL representation of Polynomial

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

struct Node {
    int coef;
    int exp;
    struct Node* next;
};

typedef struct Node Node;
```

```
void insert(Node** poly, int coef, int exp) {
    Node* temp = (Node*) malloc(sizeof(Node));
    temp->coef = coef;
    temp->exp = exp;
    temp->next = NULL;

    if (*poly == NULL) {    *poly = temp;        return;    }

    Node* current = *poly;

    while (current->next != NULL)  current = current->next;

    current->next = temp;
}
```

```
void print(Node* poly) {
    if (poly == NULL) {
        printf(“0\n");
        return;
    }
    Node* current = poly;

    while (current != NULL) {
        printf("%dx^%d", current->coef, current->exp);
        if (current->next != NULL) printf(" + ");
       current = current->next;
    }
printf("\n");
}
```

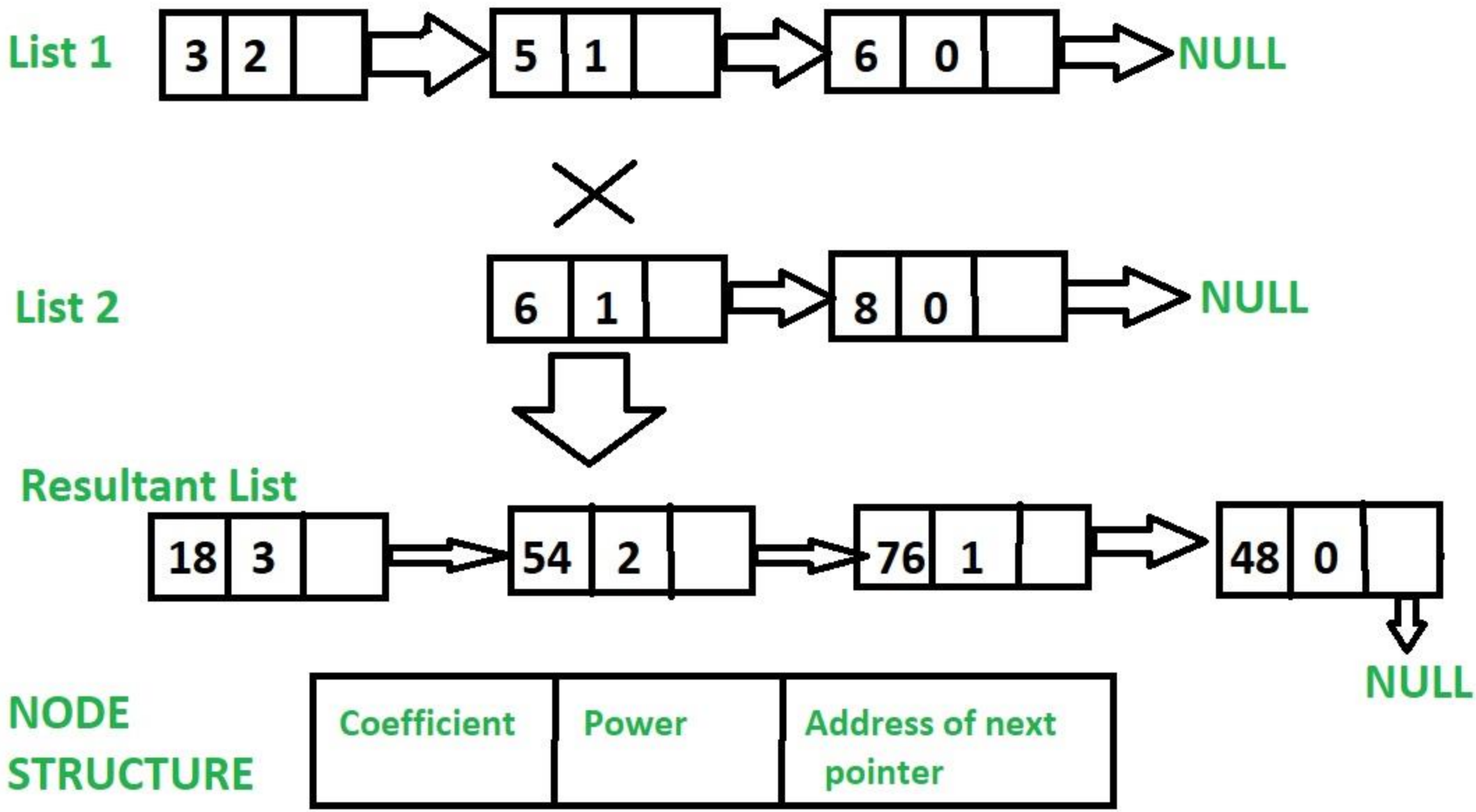
```
Input:
      1st number = 5x^2 + 4x^1 + 2x^0
      2nd number = -5x^1 - 5x^0
Output:
          5x^2-1x^1-3x^0
Input:
      1st number = 5x^3 + 4x^2 + 2x^0
      2nd number = 5x^1 - 5x^0
Output:
          5x^3 + 4x^2 + 5x^1 - 3x^0
```


List 1
5
2
4
1
2
0
NULL
List 2
5
1
5
0
NULL
Resultant List
5
2
9
1
7
0
NULL
NODE STRUCTURE
Coefficient
Power
Address of next node

```
Node* add(Node* poly1, Node* poly2) {
    Node* result = NULL;
    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->exp == poly2->exp) {
            insert(&result, poly1->coef + poly2->coef, poly1->exp);
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
            else if (poly1->exp > poly2->exp) {
            insert(&result, poly1->coef, poly1->exp);
            poly1 = poly1->next;
        }
            else {
            insert(&result, poly2->coef, poly2->exp);
            poly2 = poly2->next;
        }
    }
```

```
    while (poly1 != NULL) {
        insert(&result, poly1->coef, poly1->exp);
        poly1 = poly1->next;
    }

    while (poly2 != NULL) {
        insert(&result, poly2->coef, poly2->exp);
        poly2 = poly2->next;
    }

    return result;
}
```

```
int main()
{       int n,i,e,c; Node *first1=NULL, *first2=NULL, *first3=NULL, *first4=NULL;
        printf("how many terms in first polynomial\n";      scanf(“%d”,&n);
        for(i=0;i<n;i++)
        {
                printf(“Enter coe: “);              scanf(“%d”,&c);
                printf("Enter exp: “);              scanf(“%d”,&e);
                first1=insert(first1,c,e);
        }
        display(first1);
        printf("how many terms in second polynomial\n";   scanf(“%d”,&n);
        for(i=0;i<n;i++)
        {       printf(“Enter coe: “);              scanf(“%d”,&c);
                printf("Enter exp: “);              scanf(“%d”,&e);
                first2=a.create(first2,c,e);
        }
        print(first2);
        first3=add(first1,first2);    first4=mul(first1,first2);
        cout<<"Result of addition: ";               print(first3);
        cout<<"Result of multiplication: ";  print(first4);      }
```


List 1
3
2
5
1
6
0
NULL
List 2
6
1
8
0
NULL
Resultant List
18
3
54
2
76
1
48
0
NULL
NODE STRUCTURE
Coefficient
Power
Address of next pointer

## Examples:

**Input:** Poly1: 3x^2 + 5x^1 + 6, Poly2: 6x^1 + 8

**Output:** 18x^3 + 54x^2 + 76x^1 + 48

On multiplying each element of 1st polynomial with

elements of 2nd polynomial, we get

18x^3 + 24x^2 + 30x^2 + 40x^1 + 36x^1 + 48

On adding values with same power of x,

18x^3 + 54x^2 + 76x^1 + 48

**Input:** Poly1: 3x^3 + 6x^1 - 9, Poly2: 9x^3 - 8x^2 + 7x^1 + 2

**Output:** 27x^6 - 24x^5 + 75x^4 - 123x^3 + 114x^2 - 51x^1 - 18

```
node * node::mul(node *f1,node *f2)
{
        node *res=NULL;
        for(node *i=f1;i!=NULL;i=i->next)
                for(node *j=f2;j!=NULL;j=j->next)
                        insert(&res,i->coe*j->coe,i->exp+j->exp);
        return(res);
}
```

```
Node* multiply(Node* poly1, Node* poly2)
{
    Node *ptr1, *ptr2,*poly3;
    ptr1 = poly1;     ptr2 = poly2;
    while (ptr1 != NULL) {
        while (ptr2 != NULL) {
            int coeff, power;
            coeff = ptr1->coeff * ptr2->coeff;
            power = ptr1->power + ptr2->power;
            poly3 = insert(poly3, coeff, power);
            ptr2 = ptr2->next;
        }
        ptr2 = poly2;
        ptr1 = ptr1->next;
    }
    return poly3;
}
```

1st Polynomial:- 3x^3+6x^1-9
2nd Polynomial:- 9x^3-8x^2+7x^1+2
Resultant Polynomial:- 27x^6-24x^5+75x^4-123x^3+114x^2-51x^1-18

```
node *simplify(node *r)
{
    if(!r)     return(NULL);
     node *c,res2=NULL;
    int i,coeSum=0,maxExp=r->exp;
    for(i=maxExp;i>=0;i--){
        coeSum=0;
       for(c=r;c;c->next)  if(c->exp==i)    coeSum=coeSum+c->coe;
        create(&res2,coeSum,i);
      }//for
return(res2);
}
```