DATA STRUCTURES

UNIT-2

Polynomials

Dr G.Kalyani

Topics

- **What is Polynomial**
- Polynomial representation
- Adding of two polynomials

What is a polynomial

- A polynomial p(x) is the expression in variable x which is in the form

  $(ax^n + bx^{n-1} + \ldots . + jx + k)$,

  where a, b, c …., k fall in the category of real numbers and 'n' is non negative integer, called the degree of polynomial.

- each term in the polynomial expression consists of two parts:
  - one is the coefficient
  - other is the exponent
- <u>Example:</u>

  $10x^2 + 26x$, here 10 and 26 are coefficients and 2, 1 is its exponential value.

Topics

- What is Polynomial
- **Polynomial representation**
- Adding of two polynomials

Representation of polynomials

- **Polynomial can be represented two ways.**
  - By the use of arrays
  - By the use of Linked List

- **By the use of arrays**
  - Consider two arrays: Coefficient and Exponent
  - For n degree polynomial define the arrays of (n+1) size
  - Ex1 : $10x^4 + 6x^3 + 9x^2 + 26x + 15$
    - Coefficient array: 10, 6, 9, 26, 15
    - Exponent array: 4, 3, 2, 1, 0
  - Ex2: $6x^7 + 8x^5 + 2x^4 + 5x + 13$
    - Coefficient array: 6, 0, 8, 2, 0, 0, 5, 13
    - Exponent array: 7, 6, 5, 4, 3, 2, 1, 0

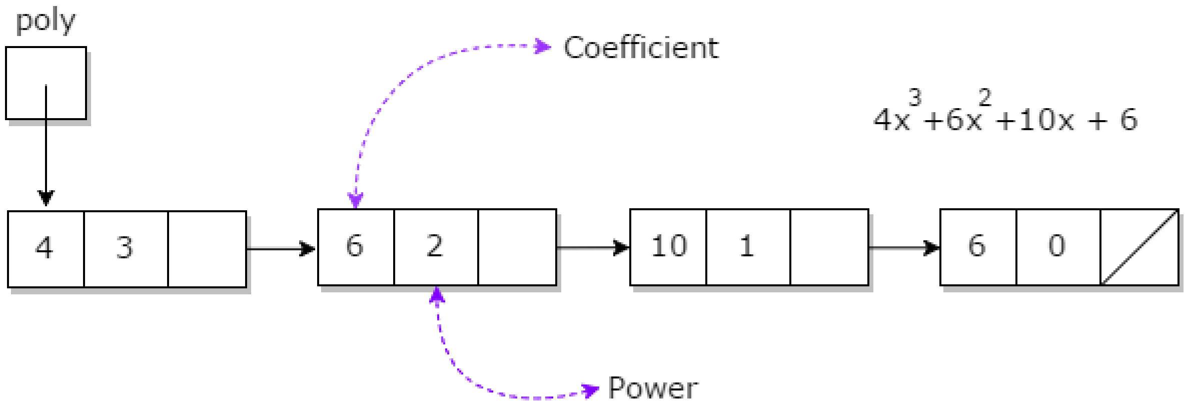
Linked list representation of polynomial

- **By the use of linked list**
  - Format of a node

Coefficient	Exponent	Link

```
struct polynode {
      int coef;
      int exp;
      struct polynode * next;
 };
```

- Example:



Topics

- What is Polynomial
- Polynomial representation
- **Adding of two polynomials**

Adding of Two polynomials

- Given two polynomial numbers represented by a linked list.
- Add these lists means add the coefficients who have same variable powers.

- **Example:**

Input:

1st polynomial = $5x^2 + 4x^1 + 2x^0$

2nd polynomial = $-5x^1 - 5x^0$

Output: $5x^2 - 1x^1 - 3x^0$

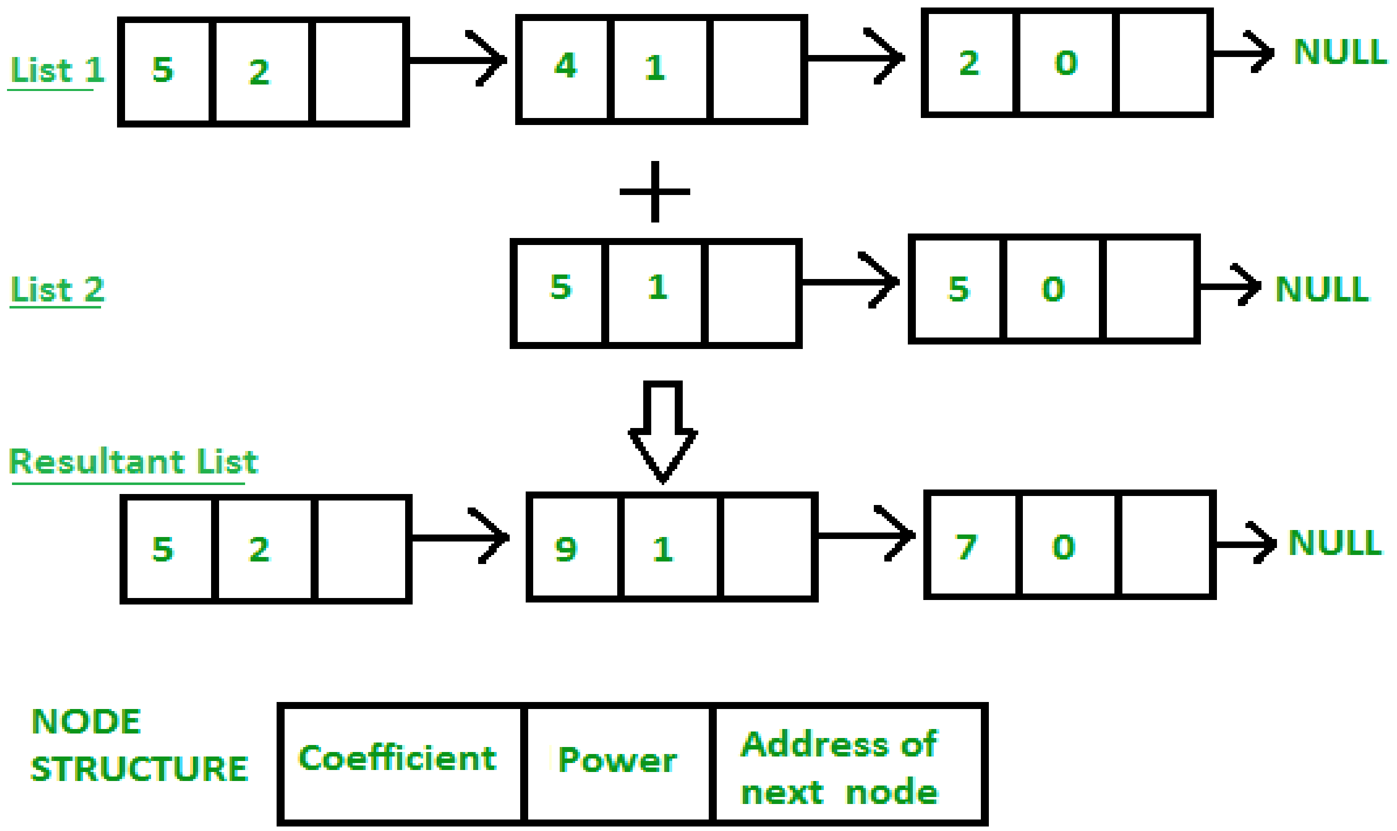
Input:

1st polynomial = $5x^3 + 4x^2 + 2x^0$

2nd polynomial = $5x^1 - 5x^0$

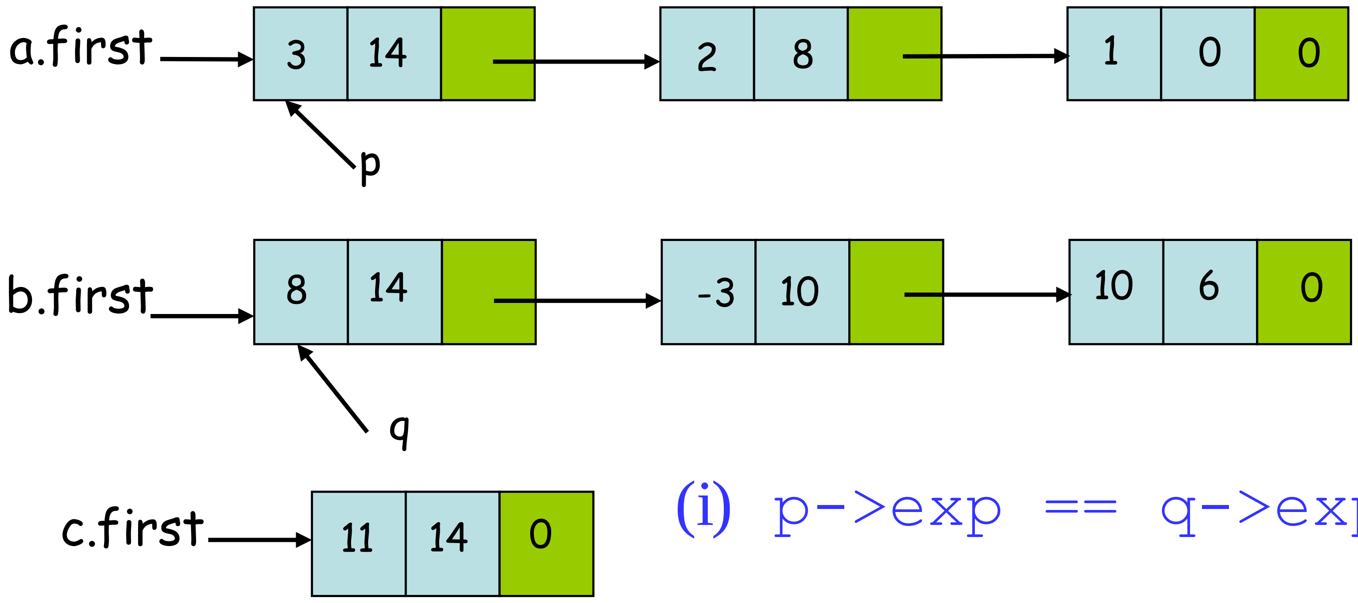
Output: $5x^3 + 4x^2 + 5x^1 - 3x^0$

Adding of Two polynomials



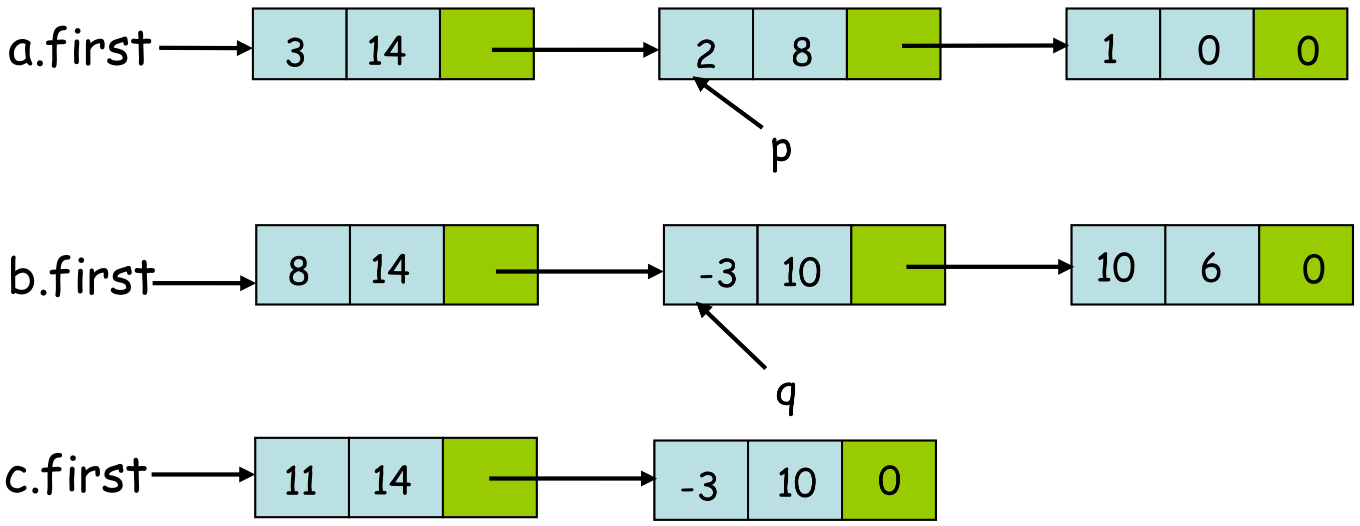
Adding of Two polynomials

- Example of adding two polynomials a and b



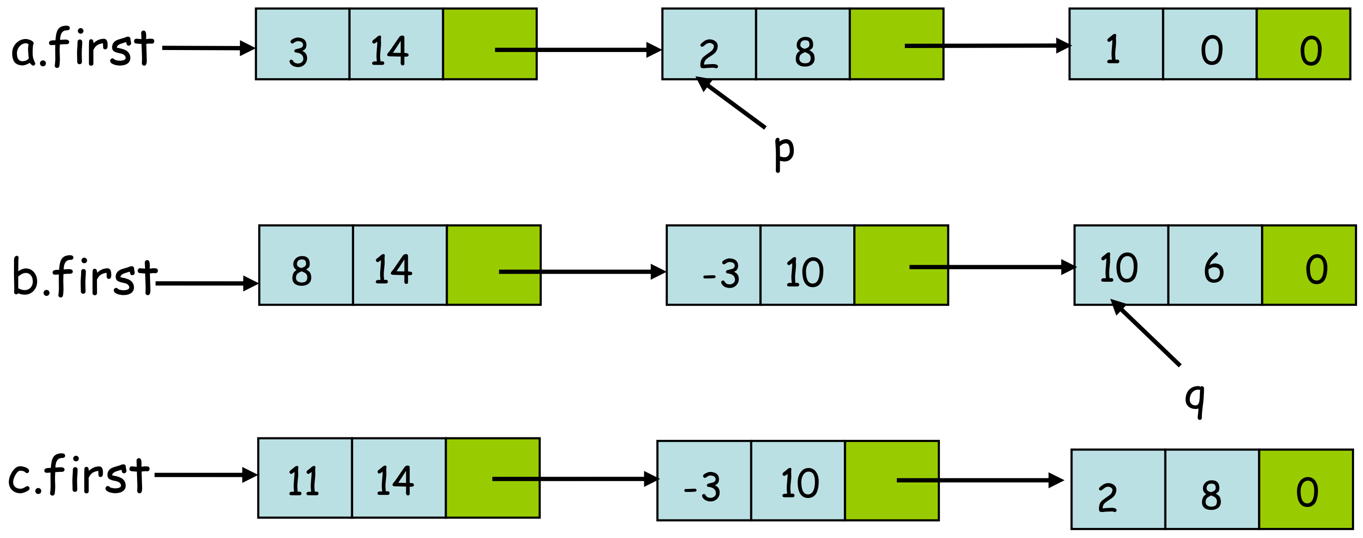
(i) `p->exp == q->exp`

Adding of Two polynomials



(ii) `p->exp < q->exp`

Adding of Two polynomials



(iii) `p->exp > q->exp`

```
Algorithm add_poly(struct node *result, struct node * poly1, struct node * poly2)
{
    while(poly1 && poly2)
    {
      if (poly1->pow > poly2->pow)
       {

       }
     else if (poly1->pow < poly2->pow)
          {

          }
     else  {

          }
    }// while loop close
```

```
  //Loop while either of the
linked lists has value
   if(poly1)
    {

    }
  if(poly2)
    {

    }
}
```

```
Algorithm add_poly(struct node *result, struct node * poly1, struct node * poly2)
{
    struct node * new_node;
    new_node = (struct node *) malloc(sizeof(struct node*));
    new_node->next = NULL;
    result = new_node;
    while(poly1 && poly2)  {
       if (poly1->pow > poly2->pow) {
          new_node->pow = poly1->pow;
          new_node->coeff = poly1->coeff;
          poly1 = poly1->next
                                         }
       else if (poly1->pow < poly2->pow) {
                new_node->pow = poly2->pow;
                new_node->coeff = poly2->coeff;
                poly2 = poly2->next;
                                              }
        else  {
                new_node->pow = poly1->pow;
                new_node->coeff = poly1->coeff + poly2->coeff;
                poly1 = poly1->next;
                poly2 = poly2->next;
             }
       if(poly1 && poly2)
       {
        new_node->next=(struct node *) malloc
                                           (sizeof(struct node*));
         new_node= new_node->next;
          new_node->next = NULL;
       }
    }// while loop close
    
 //Loop while either of the linked lists has value
while(poly1 || poly2) {
 new_node->next = (struct node *)
                                   malloc(sizeof(struct node*));
 new_node = tmp_node->next;
 new_node->next = NULL;

  if(poly1) {
      new_node->pow = poly1->pow;
     new_node->coeff = poly1->coeff;
      poly1 = poly1->next;
            }
 if(poly2) {
     new_node->pow = poly2->pow;
     new_node->coeff = poly2->coeff;
     poly2 = poly2->next;
           }
   }// while loop close

   write Addition Complete;
}
```

Reading the polynomial

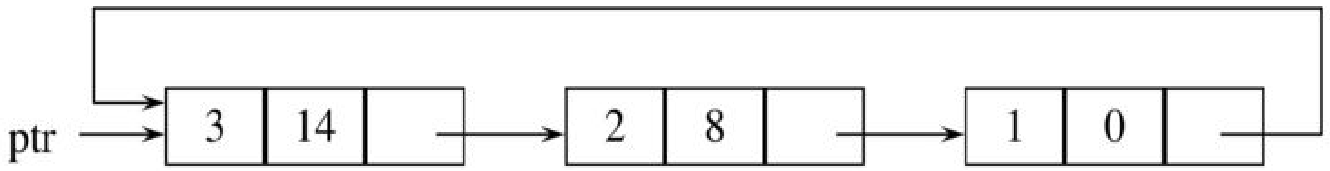
```
Algorithm readPolynomial(struct Node** poly)
{
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        *poly = temp;
        do{
                read Coeffecient;
                read Exponent;
                temp->coeff = coeff;
                temp->pow = exp;
                temp-> next = NULL;
                write Have more terms? 1 for y and 0 for no;
                read cont;
                if(cont)
                {
                        temp->next = (struct Node*)malloc(sizeof(struct Node));
                        temp = temp->next;
                        temp->next = NULL;
                }
        }while(cont);
}
```

Display the polynomial

```
Algorithm  displayPolynomial(struct Node* poly)
{
        write Polynomial expression is:;
        while(poly  != NULL)
        {
                write poly->coeff ^ poly->pow;
                poly = poly->next;
                if(poly != NULL)
                        write "+";
        }
}
```

Circular list representation of polynomials

- Representing polynomials as circularly linked lists
  - We can free all the nodes more efficiently if we modify our list structure so that the link field of the last node points to the first node in the list, called **circular list**.
  - A singly linked list in which the last node has a null link is called a **chain**.



- Erase a circular list in a fixed amount of time