Application of Linked Lists:
Polynomial Representation and Addition

Polynomials

- Representing Polynomials as a Singly Linked Lists
  - The manipulation of symbolic polynomials(list processing).
  - In general, we want to represent the polynomial:

$$A(x) = a_{m-1}x^{e_{m-1}} + \cdots + a_0 x^{e_0}$$

    - Where the a_i are nonzero coefficients and the e_i are nonnegative integer exponents such that

$$e_{m-1} > e_{m-2} > \ldots > e_1 > e_0 \geqq 0 .$$

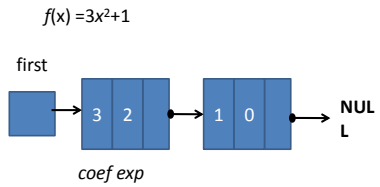
  - We will represent each term as a **node** containing **coefficient** and **exponent** fields, as well as a **pointer** to the next term.

Polynomial Representation

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

struct poly_node *first=NULL;
```


Polynomial Representation

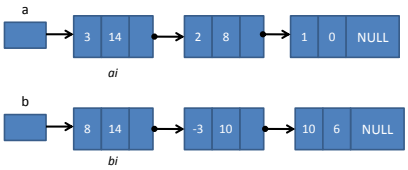


Adding Polynomial

- Example
  - Consider the following two polynomials:

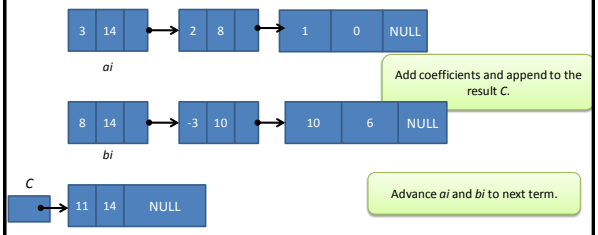
$a(x) = 3x^{14}+2x^{8}+1$

$b(x) = 8x^{14}-3x^{10}+10x^{6}$



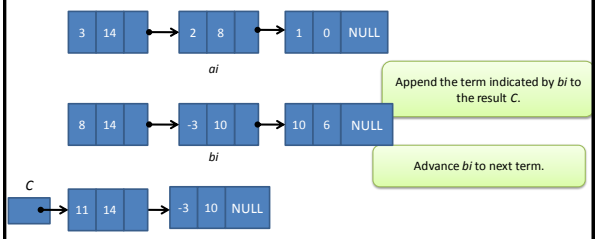
Case 1

- *ai->exp* == *bi->exp*



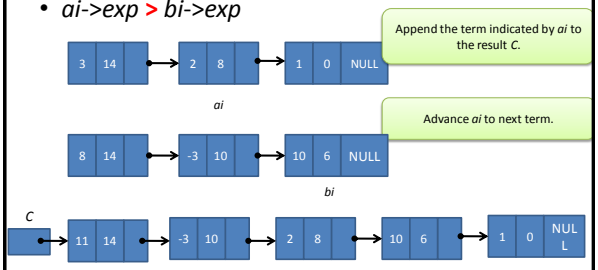
Case 2

- *ai->exp* < *bi->exp*



Case 3

- *ai->exp* > *bi->exp*



Algorithm of Add()

```
Struct poly_node * addpoly(struct  poly_node *a, struct
                                   poly_node * b)
{
     struct poly_node *c,*atemp,*btemp; int sum=0;
     atemp=a;
     btemp= b;
```



```
while (atemp!= NULL && btemp != NULL)
    {
        if (atemp->exp == btemp->exp) {
            sum = atemp->coef + btemp->coef;
        addatend(c,sum, atemp->exp);    // Add a node with at the
end of the  Resultant polynomial C
            atemp = atemp->next;
            btemp= btemp->next;
        }
        else if (atemp->exp < btemp->exp) {
            addatend( c,btemp->coef, btemp->exp);
            btemp = btemp-> next;
        }
       else if (atemp->exp > btemp->exp) {
            addatend( c,atemp->coef, atemp->exp);
            atemp = atemp-> next;
        }
    }
```


Algorithm of Add()

```
for (; atemp!= NULL; atemp= atemp->next)
        addatend(c,atemp->coef, atemp->exp);
for (; btemp!= NULL; btemp= btemp-> next)
        addatend(c, btemp->coef, btemp->exp);
 return c;
}
```


Function to add a node at the end

```
Addatend(struct poly_node *a,int c,int e)
{struct poly_node *temp, *newNode;
 newNode=(struct poly_node *)malloc(sizeof(struct
   poly_node));
 newNode->coef=c;
newNode->exp=e;
newNode->next=NULL;
```

Function to add a node at the end

```
if(a==NULL) {a=newNode;}
else
{ temp=a;
  while(temp->next!=NULL)
    {temp=temp->next;}
   temp->next=newNode;}
}
```