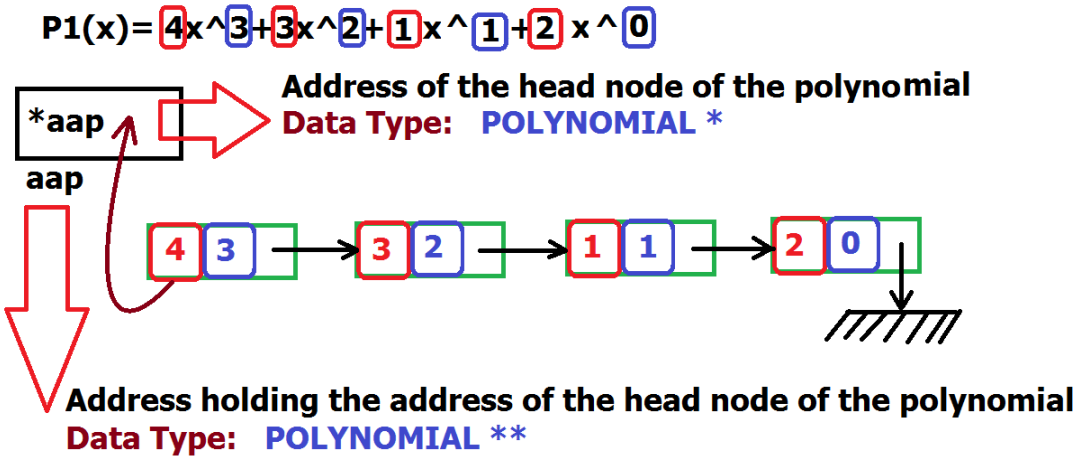
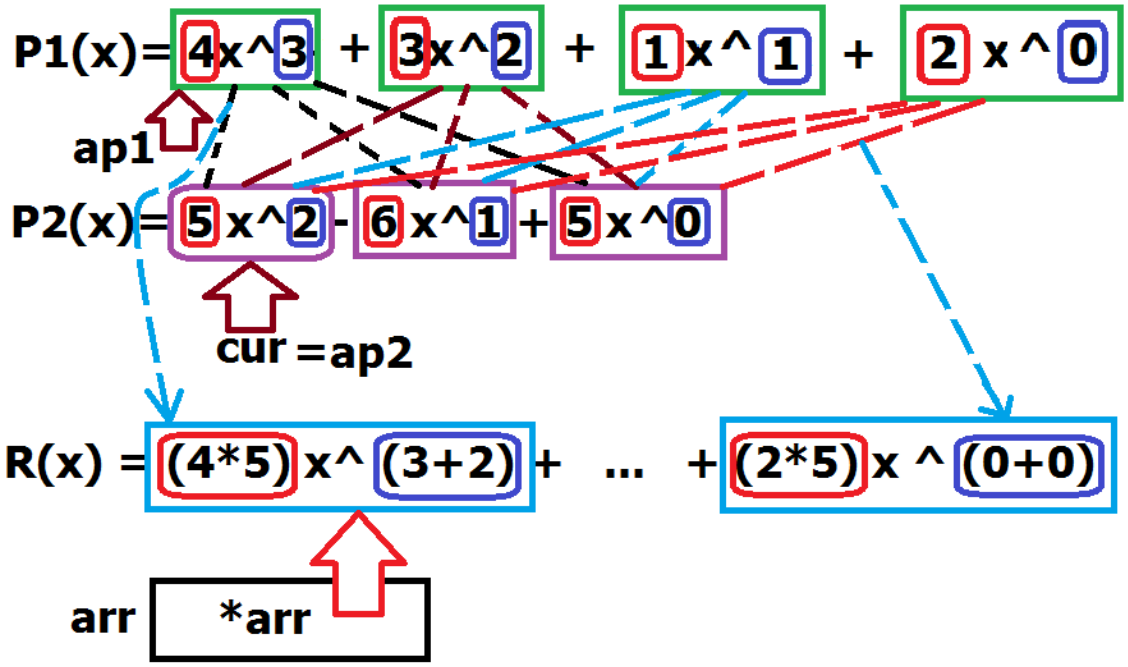
Important Functions Related to Polynomial Addition and Multiplication Using Linked List

Dr. Rahul Das Gupta




```
typedef struct polynomial
{
      int coefficient, exp;
      struct polynomial * nxt;
}POLYNOMIAL;
```



Multiplication of two polynomials represented by Linked Lists i.e., R(x)=P1(x)*P2(x)

==========*/

/**********

Important points to be noted:

#1: The **first linked list for the Polynomial P1(x)** need to be travel **form starting node to the last node just for one time**.

#2: For each node of the first linked list [for the Polynomial P1(x)], the **second linked list for the Polynomial P2(x)** need to be travel form **starting node to the last node**.

#3: For each elementary multiplication between the nodes of **first and second linked lists [for Polynomials P1(x) and P2(x) respectively]** a **new node is inserted in the descending order of exponent in the third linked list holding the result of multiplication**.

**********/

```
void initialize(POLYNOMIAL **aap)
{
        *aap=NULL;
}
```

/* Insertion of nodes in the descending order of exponent without duplication. */

```
void sinsert_dwd (POLYNOMIAL **aap, int c, int e)
{
    POLYNOMIAL *cur,*prv,*t;
    if(c==0) return;

    for(cur=*aap, prv=NULL;cur && e<cur->exp; cur=cur->nxt)
        prv=cur;
    if(cur && e == cur->exp)
    {
        (cur->coefficient)+=c;
        if (cur->coefficient == 0)
        /*   A node is created with Zero Coefficient (as for example 0x^2).  */
        {
            if(prv)
                prv->nxt=cur->nxt;
            else
                *aap=cur->nxt;
            free(cur);
        }
        return;
    }
   t=(node *)malloc (sizeof(POLYNOMIAL));
   t->coefficient=c;
   t->exp=e;
   t->nxt=cur;

    if(prv)
        prv->nxt=t;
    else
       *aap=t; /*when it is the first node*/

}
```

```
void multiplication (POLYNOMIAL *ap1, POLYNOMIAL *ap2, POLYNOMIAL **aar)
/*================================================================
It is important to note that we just need to read the two linked lists, but we need to
create a new third linked list to hold the result of the multiplication.
================================================================*/
{
    POLYNOMIAL *cur;
    initialize(aar);

    for(;ap1;ap1=ap1->nxt)
        for(cur=ap2;cur;cur=cur->nxt)
          insert(aar, (ap1->coefficient)*(cur-> coefficient), (ap1->exp)+(cur->exp));
/*================================================================
```

For each elementary multiplication between the nodes of **first and second linked lists [for Polynomials P1(x) and P2(x) respectively]** a **new node is inserted in the descending order of exponent in the third linked list holding the result of multiplication**.

```
================================================================*/
}
```

```
void  poly_add (POLYNOMIAL *aph1, POLYNOMIAL *aph2, POLYNOMIAL **aar)

{
  initialize(aar);
  for( ;aph1; aph1=aph1->nxt)
          sinsert_dwd(aar,aph1->coefficient, aph1->exp);
  for( ;aph2;aph2=aph2->nxt)
          sinsert_dwd(aar,aph2->coefficient, aph2->exp);
}


void input(POLYNOMIAL **aah)
{
   int c,e;
   char ans;
   initialize(aah);
  do
  {
       printf("\nEnter the coefficient: ");
       scanf("%d", &c);
       printf("Enter the exponent: ");
```

```
        scanf("%d", &e);
        insert(aah,c,e);
        printf("\n Do you want to continue?(Y/N)\n");
        scanf("%*c %c", &ans);
    }while(ans!='N' && ans!='n');
}

void display(POLYNOMIAL *ap)
{
     POLYNOMIAL *cur=ap;
     printf("\n\t %d x^%d",cur->coefficient,cur->exp);
     for(cur=ap->nxt; cur; cur=cur->nxt)
     {
          if(cur->c==0)
                    continue;
          else if(cur->coefficient>0)
                     printf("+%d x^%d", cur->coefficient, cur->exp);
          else
                     printf("%d x^%d", cur->coefficient, cur->exp);
      }
}
```