Polynomial manipulation

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

struct Node {
   int coeff;
   int exp;
   struct Node * next;
}* poly1 = NULL,*poly2=NULL,*result=NULL;

void create1() {
   struct Node * t, * last = NULL;
   int num, i;

   printf("Enter number of terms: ");
   scanf("%d", & num);
   printf("Enter each term with coeff and exp:\n");

   for (i = 0; i < num; i++) {
      t = (struct Node * ) malloc(sizeof(struct Node));
      scanf("%d%d", & t -> coeff, & t -> exp);
      t -> next = NULL;
      if (poly1 == NULL) {
         poly1 = last = t;
      } else {
         last -> next = t;
         last = t;
      }
   }
}

void create2() {
   struct Node * t, * last = NULL;
   int num, i;

   printf("Enter number of terms: ");
   scanf("%d", & num);
   printf("Enter each term with coeff and exp:\n");

   for (i = 0; i < num; i++) {
      t = (struct Node * ) malloc(sizeof(struct Node));
      scanf("%d%d", & t -> coeff, & t -> exp);
      t -> next = NULL;
      if (poly2 == NULL) {
         poly2 = last = t;
      } else {
         last -> next = t;
         last = t;
      }
   }
}


void Display(struct Node * p) {
   printf("(%dx^%d) ", p -> coeff, p -> exp);
   p = p -> next;

   while (p) {
```

```
        printf("+ (%dx^%d)", p -> coeff, p -> exp);
        p = p -> next;
    }
    printf("\n");
}

void normalize(){
    struct Node *ptr=result;
    while(ptr){
        struct Node *temp=ptr;
        while(temp->next){
            struct Node*erase;
            if(ptr->exp==temp->next->exp){
                ptr->coeff=ptr->coeff+temp->next->coeff;
                erase=temp->next;
                temp->next=temp->next->next;
                free(erase);
            }
            temp=temp->next;
        }
        ptr=ptr->next;
    }
}

void add(struct Node * p1, struct Node * p2) {
    struct Node * t, * last = NULL;
    int num1,num2;
    int p;
    if(p1->exp>p2->exp)
        p=p1->exp;
    else
        p=p2->exp;

    while (p) {
        p--;
        if(p1!=NULL && p2!= NULL) {
            t = (struct Node * ) malloc(sizeof(struct Node));
            if(p1->exp==p2->exp) {
                num1=p1->exp;
                num2=p1->coeff+p2->coeff;
                p1 = p1 -> next;
                p2 = p2 -> next;
            }
            else if(p1->exp>p2->exp) {
                num1=p1->exp;
                num2=p1->coeff;
                p1 = p1 -> next;
            }
            else if(p1->exp<p2->exp) {
                num1=p2->exp;
                num2=p2->coeff;
                p2 = p2 -> next;
            }
            t->exp=num1;
            t->coeff=num2;

            t -> next = NULL;
            if (result == NULL) {
                result = last = t;
            } else {
                last -> next = t;
```

```
            last = t;
        }
    }
    if(p1!=NULL && p2==NULL) {
        t = (struct Node * ) malloc(sizeof(struct Node));
        num1=p1->exp;
        num2=p1->coeff;
        p1 = p1 -> next;
        t->exp=num1;
        t->coeff=num2;

        t -> next = NULL;
        if (result == NULL) {
            result = last = t;
        } else {
            last -> next = t;
            last = t;
        }
    }
    if(p1==NULL && p2!=NULL) {
        t = (struct Node * ) malloc(sizeof(struct Node));
        num1=p2->exp;
        num2=p2->coeff;
        p2 = p2 -> next;
        t->exp=num1;
        t->coeff=num2;

        t -> next = NULL;
        if (result == NULL) {
            result = last = t;
        } else {
            last -> next = t;
            last = t;
        }
    }
  }
  normalize(result);
  Display(result);
  result=NULL;
}

void sub(struct Node * p1, struct Node * p2) {
  struct Node * t, * last = NULL;
  int num1,num2;
  int p;
  if(p1->exp>p2->exp)
    p=p1->exp;
  else
    p=p2->exp;

  while (p) {
    p--;
    if(p1!=NULL && p2!= NULL) {
        t = (struct Node * ) malloc(sizeof(struct Node));
        if(p1->exp==p2->exp) {
            num1=p1->exp;
            num2=p1->coeff-p2->coeff;
            p1 = p1 -> next;
            p2 = p2 -> next;
        }
        else if(p1->exp>p2->exp) {
```

```
            num1=p1->exp;
            num2= - p1->coeff;
            p1 = p1 -> next;
        }
        else if(p1->exp<p2->exp) {
            num1=p2->exp;
            num2= - p2->coeff;
            p2 = p2 -> next;
        }
        t->exp=num1;
        t->coeff=num2;

        t -> next = NULL;
        if (result == NULL) {
            result = last = t;
        } else {
            last -> next = t;
            last = t;
        }
    }
    if(p1!=NULL && p2==NULL) {
        t = (struct Node * ) malloc(sizeof(struct Node));
        num1=p1->exp;
        num2=p1->coeff;
        p1 = p1 -> next;
        t->exp=num1;
        t->coeff=num2;

        t -> next = NULL;
        if (result == NULL) {
            result = last = t;
        } else {
            last -> next = t;
            last = t;
        }
    }
    if(p1==NULL && p2!=NULL) {
        t = (struct Node * ) malloc(sizeof(struct Node));
        num1=p2->exp;
        num2= - p2->coeff;
        p2 = p2 -> next;
        t->exp=num1;
        t->coeff=num2;

        t -> next = NULL;
        if (result == NULL) {
            result = last = t;
        } else {
            last -> next = t;
            last = t;
        }
    }
  }
  normalize(result);
  Display(result);
  result=NULL;
}

void mul(struct Node*p1,struct Node*p2) {
  int i=1;
  struct Node *t, *last=NULL;
```

```
    while(p1) {
        struct Node*temp=p2;
        while(temp) {
            t=malloc(sizeof(struct Node));
            t->coeff=(temp->coeff) * (p1->coeff);
            t->exp=temp->exp+p1->exp;
            if (result == NULL) {
                result = last = t;
            } else {
                last -> next = t;
                last = t;
            }
            temp=temp->next;
        }
        p1=p1->next;
    }
    normalize(result);
    Display(result);
    result=NULL;
}

int main() {
    int x;
    printf("enter first polynomial:\n");
    create1();
    Display(poly1);
    create2();
    Display(poly2);
    printf("\nADD: ");
    add(poly1,poly2);
    printf("\nSUB: ");
    sub(poly1,poly2);
    printf("\nMUL: ");
    mul(poly1,poly2);
    return 0;
}
```

OUTPUT

```
Enter the values for first polynomial :
Enter the coefficient : 2
Enter the power : 2
Enter 1 to continue : 1
Enter the coefficient : 6
Enter the power : 1
Enter 1 to continue : 1
Enter the coefficient : 5
Enter the power : 0
Enter 1 to continue : 0
The polynomial equation is : 2x^2+6x^1+5x^0
Enter the values for second polynomial :
Enter the coefficient : 3
Enter the power : 2
Enter 1 to continue : 1
Enter the coefficient : -2
Enter the power : 1
Enter 1 to continue : 1
Enter the coefficient : -1
Enter the power : 0
Enter 1 to continue : 0
The polynomial equation is : 3x^2-2x^1-1x^0
```

The polynomial equation addition result is : 5x^2+4x^1+4x^0