10- Polynomial Addition

Aim

Polynomial Addition using arrays.

Algorithms

I Struct polynom

1. START
2. ~~int~~ create coeff & power variable of int datatype.
3. STOP

II create global arrays ~~variable~~ of struct datatype such as p1[20], p2[20], sum[20];

III void insertpoly (struct polynom poly[], int len)

1. START
2. for (int i = 0 to len)
   1. Get the exponent and coefficient from user and store it in poly[]
3. End for.
4. STOP

IV int polyadd()

1. START
2. int i=0, l=0, m=0

   // i = count of sum poly
   // l = count of 1st poly
   // m = count of 2nd poly.

3. while((l < len(p1)) and (m < len(p2))) then
   1. if(p1[l].powr > p2[k].powr)
      1. sum[i].powr = p1[l].powr;
      2. sum[i].coeff = p1[l].coeff;
      3. ~~Increment~~ l++
   2. Else if (p1[l].powr < p2[l].powr)
      1. sum[i].powr = p2[k].powr;
      2. sum[i].~~coeff~~ = p2[k].pcoeff;
      3. k++
   3. Else
      1. sum[i].powr = p1[l].powr;
      2. sum[i].coeff = p1[l].coeff + p2[k].coeff;
      3. l++ and k++
   4. ~~end~~ End If
   5. i++

4. End While
5. While (l < len(p1)) Then
    1. sum[i].powr = p1[l].powr;
    2. sum[i].coeff = p1[l].coeff ~~p2[k].coeff~~;
    3. l++ and i++
6. End While
7. While (k < len(p2)) Then
    1. Sum[i].powr = p2[k].powr;
    2. sum[i].coeff = p2[k].coeff;
    3. k++ and i++
8. End While
9. ~~Stop~~ return i
10. STOP

V void disp (int ~~--~~ sizeOfsum)

1. START
2. print ("Polynomial A:")
3. for (int i=0 to len(p1)) {
    1. print ("~~+~~", p1[i].coeff, "(X^", p1[i].coeff"))
4. for (int i=0 to len(p2))

4. print "", p2[i].coeff, "(X ^ ", p2[i] power, ")");

5. for (int i = 0 to sizeOfsum)

   1. printf ("%d (X ^ %d)", gsum[i].coeff, gsum[i].power);

6. STOP

VI int main()

1. START
2. print ("Polynomial 1 → \n");
3. insertpoly (p1, len(p1))
4. print ("Polynomial 2 → \n");
5. insertpoly (p2, len(p2))
6. int sizeOfsum = polyadd();
7. disp (sizeOfsum);
8. STOP

Output

Obtained & verified

```
#include <stdio.h>

#define n 3
#define m 4
int l=0,k=0;

struct polynom{
        int coeff;
        int powr;
} p1[20] ,p2[20] ,sum[20];

void sort(struct polynom poly[] ,int len){
        int t1,t2;
        for(int i=0;i<len-1;i++){
                for(int j=0;j<len-i-1;j++){
                        if(poly[j].powr < poly[j+1].powr){

                                t1=poly[j].powr;
                                t2=poly[j].coeff;

                                poly[j].powr =poly[j+1].powr;
                                poly[j].coeff=poly[j+1].coeff;

                                poly[j+1].powr = t1;
                                poly[j+1].coeff=t2;
                        }
                }
        }
}

void insertpoly(struct polynom poly[],int len){
        int co,exp;
        for(int i=0 ;i<len ;i++){
                printf("\texponent: ");
                scanf("%d",&exp);
                poly[i].powr=exp;
                printf("\tcoefficient of X^%d: ",exp);
                scanf("%d",&co);
                poly[i].coeff=co;
        }
}
int polyadd(){
        int i=0,sc=0;
        while(l<n && k<m){
                if(p1[l].powr>p2[k].powr){
                        sum[i].powr = p1[l].powr;
                        sum[i].coeff=p1[l].coeff;
                        l++;sc++;
                }
                else if(p1[l].powr<p2[k].powr){
```

```
                        sum[i].powr = p2[k].powr;
                        sum[i].coeff=p2[k].coeff;
                        k++;sc++;
                }
                else{
                        sum[i].powr = p1[l].powr;
                        sum[i].coeff= p1[l].coeff + p2[k].coeff;
                        l++;
                        k++;
            sc++;
                }
                i++;
        }
        while(l<n){
                sum[sc].powr=p1[l].powr;
                sum[sc].coeff=p1[l].coeff;
                l++;sc++;
        }
        while(k<m){
                sum[sc].powr=p2[k].powr;
                sum[sc].coeff=p2[k].coeff;
                k++;sc++;
        }
  return sc;
}

void disp(int sumc){
  int i=0;
        printf("Poly A: ");
        for(i=0 ;i<n-1 ;i++){
                printf("%d(X^%d) +",p1[i].coeff,p1[i].powr);
        }
        printf("%d(X^%d)\n",p1[i].coeff,p1[i].powr);

        printf("Poly B: ");
        for(i=0 ;i<m-1 ;i++){
                printf("%d(X^%d) +",p2[i].coeff,p2[i].powr);
        }
        printf("%d(X^%d)\n",p2[i].coeff,p2[i].powr);

        printf("\nSum: ");
        for(i=0 ;i<sumc-1 ;i++){
                printf("%d(X^%d) +",sum[i].coeff,sum[i].powr);
        }
        printf("%d(X^%d)\n",sum[i].coeff,sum[i].powr);


}

int main(){
```

```
  printf("*****Polynomial 1***** \n");
        insertpoly(p1,n);
        sort(p1,n);
  printf("*****Polynomial 2***** \n");
        insertpoly(p2,m);
        sort(p2,m);
        int sumc=polyadd();
        disp(sumc);

}
```

```
*****Polynomial 1*****
    exponent: 1
    coefficient of X^1: 1
    exponent: 2
    coefficient of X^2: 2
    exponent: 3
    coefficient of X^3: 3
*****Polynomial 2*****
    exponent: 1
    coefficient of X^1: 1
    exponent: 2
    coefficient of X^2: 2
    exponent: 3
    coefficient of X^3: 3
    exponent: 4
    coefficient of X^4: 4
Poly A: 3(X^3) +2(X^2) +1(X^1)
Poly B: 4(X^4) +3(X^3) +2(X^2) +1(X^1)

Sum: 4(X^4) +6(X^3) +4(X^2) +2(X^1)
```