22/11/2020

**Experiment No: 15**

**POLYNOMIAL USING LINKED LIST**

**AIM:**

Write a program to read two polynomials and store them using linked list. Calculate the sum and product and display the first polynomial, second polynomial and the resultant polynomial.

**DATA STRUCTURES USED:**

Linked List

**ALGORITHM:**

Algorithm POLYNOMIAL\_ADDITION()

1. Pptr =PHEADER->LINK
2. Qptr=QHEADER->LINK
3. RHEADER=GetNode(NODE)
4. RHEADER->LINK=NULL
5. RHEADER->Coeff=NULL
6. RHEADER->Exp=NULL
7. Rptr=RHEADER
8. While(Pptr!=NULL and Qptr!=NULL)
9. Case : Pptr->Exp=Qptr->Exp
10. new=GetNode(NODE)
11. Rptr->LINK=new
12. Rptr=new
13. Rptr->Coeff=Pptr->Coeff+Qptr->Coeff
14. Rptr->Exp=Pptr->Exp
15. Rptr->LINK=NULL
16. Pptr=Pptr->LINK
17. Qptr=Qptr->LINK
18. Case : Pptr->Exp>Qptr->Exp
19. new=GetNode(NODE)
20. Rptr->LINK=new
21. Rptr=new
22. Rptr->Coeff=Pptr->Coeff
23. Rptr->Exp=Pptr->Exp
24. Rptr->LINK=NULL
25. Pptr=Pptr->LINK
26. Case : Pptr->Exp<Qptr->Exp
27. new=GetNode(NODE)
28. Rptr->LINK=new
29. Rptr=new
30. Rptr->Coeff=Qptr->Coeff
31. Rptr->Exp=Qptr->Exp
32. Rptr->LINK=NULL
33. Qptr=Qptr->LINK
34. EndWhile
35. If(Pptr!=NULL and Qptr=NULL)
36. While(Pptr!=NULL)
37. new=GetNode(NODE)
38. Rptr->LINK=new
39. Rptr=new
40. Rptr->Coeff=Pptr->Coeff
41. Rptr->Exp=Pptr->Exp
42. Rptr->LINK=NULL
43. Pptr=Pptr->LINK
44. EndWhile
45. EndIf
46. If(Pptr=NULL and Qptr!=NULL)
47. While(Qptr!=NULL)
48. new=GetNode(NODE)
49. Rptr->LINK=new
50. Rptr=new
51. Rptr->Coeff=Qptr->Coeff
52. Rptr->Exp=Qptr->Exp
53. Rptr->LINK=NULL
54. Qptr=Qptr->LINK
55. EndWhile
56. EndIf

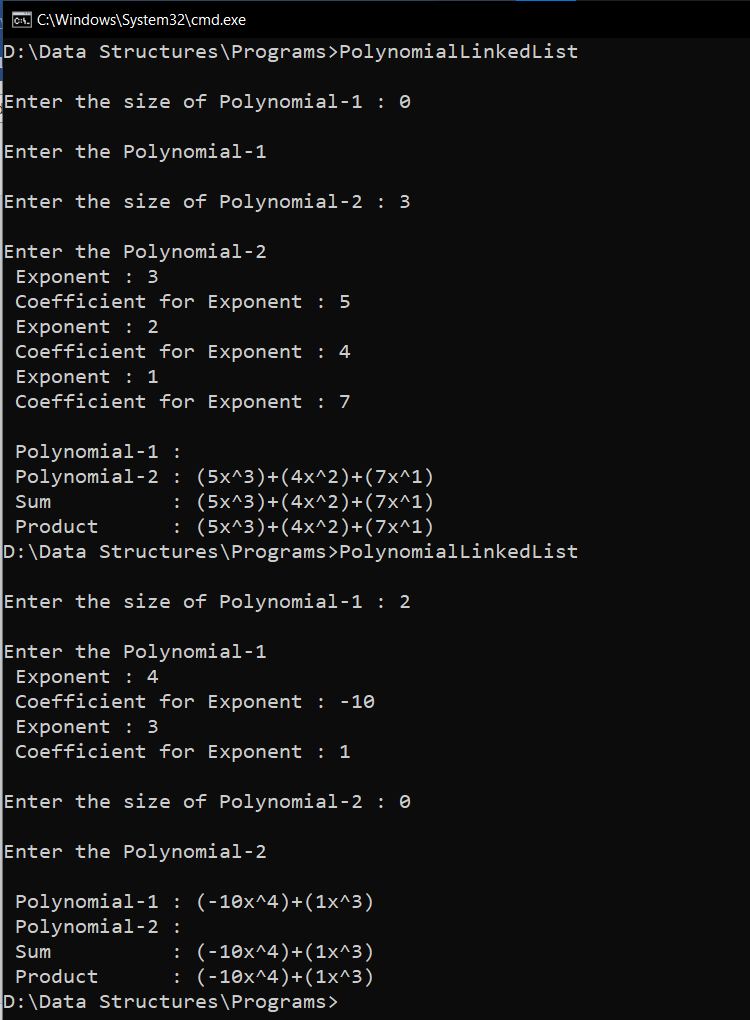
Algorithm POLYNOMIAL\_MULTIPLICATION()

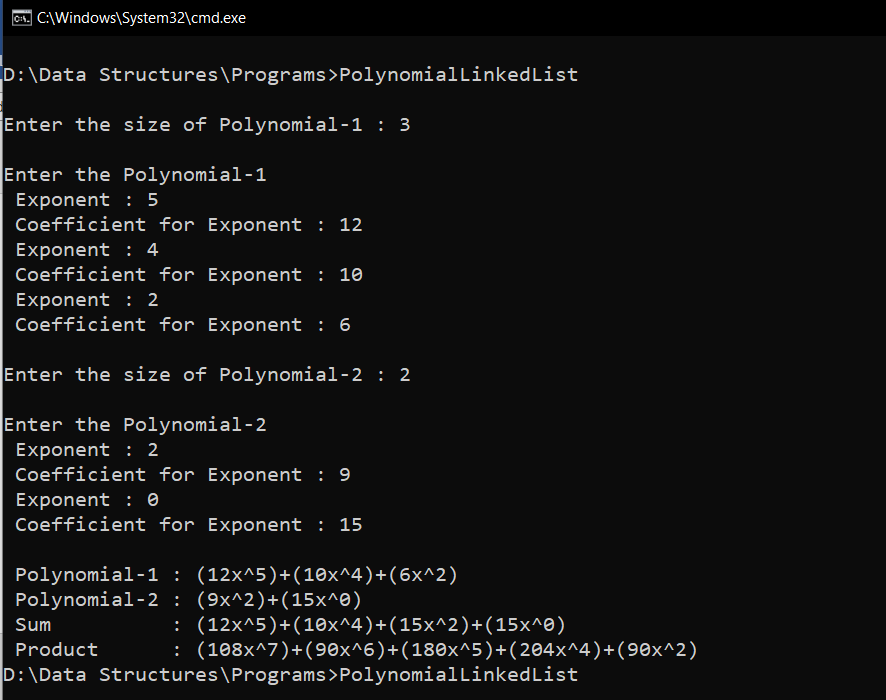
1. Pptr =PHEADER->LINK
2. Qptr=QHEADER->LINK
3. RHEADER=GetNode(NODE)
4. RHEADER->LINK=NULL
5. RHEADER->Coeff=NULL
6. RHEADER->Exp=NULL
7. If(Pptr!=NULL and Qptr=NULL)
8. While(Pptr!=NULL)
9. new=GetNode(NODE)
10. Rptr->LINK=new
11. Rptr=new
12. Rptr->Coeff=Pptr->Coeff
13. Rptr->Exp=Pptr->Exp
14. Rptr->LINK=NULL
15. Pptr=Pptr->LINK
16. EndWhile
17. Else If(Pptr=NULL and Qptr!=NULL)
18. While(Qptr!=NULL)
19. new=GetNode(NODE)
20. Rptr->LINK=new
21. Rptr=new
22. Rptr->Coeff=Qptr->Coeff
23. Rptr->Exp=Qptr->Exp
24. Rptr->LINK=NULL
25. Qptr=Qptr->LINK
26. EndWhile
27. Else
28. While(Pptr!=NULL)
29. Qptr=QHEADER->LINK
30. While(Qptr!=NULL)
31. new=GetNode(NODE)
32. Rptr->LINK=new
33. Rptr=new
34. Rptr->Coeff=Pptr->Coeff\*Qptr->Coeff
35. Rptr->Exp=Pptr->Exp+Qptr->Exp
36. Rptr->LINK=NULL
37. Qptr=Qptr->LINK
38. EndWhile
39. Pptr=Pptr->LINK
40. EndWhile
41. EndIf
42. SORT() //Sort According To Exp and Combine the Add Coeff of same Exp

**PROGRAM:**

#include<stdio.h>  
#include<stdlib.h>  
struct node{  
 int coeff;  
 int exp;  
 struct node \*link;  
};  
  
void insert(struct node\* header){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 printf(" Exponent : ");  
 scanf("%d",&new->exp);  
 printf(" Coefficient for Exponent : ");  
 scanf("%d",&new->coeff);  
 new->link=NULL;  
 if(new==NULL){  
 printf("\nMEMORY Underflow\n");  
 }else{  
 if(header->link==NULL){  
 header->link=new;  
 }else{  
 struct node\* ptr=header;  
 while(ptr->link!=NULL){  
 ptr=ptr->link;  
 }  
 ptr->link=new;  
 }  
 }  
}  
void sum(struct node\* header1,struct node\* header2,struct node\* header3){  
 struct node\* ptr1=header1->link;  
 struct node\* ptr2=header2->link;  
 struct node\* ptr3=header3;  
 while(ptr1!=NULL && ptr2!=NULL){  
 if(ptr1->exp==ptr2->exp){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr3->link=new;  
 ptr3=new;  
 ptr3->coeff=ptr1->coeff+ptr2->coeff;  
 ptr3->exp=ptr1->exp;  
 ptr3->link=NULL;  
 ptr1=ptr1->link;  
 ptr2=ptr2->link;  
 }else if(ptr1->exp>ptr2->exp){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr3->link=new;  
 ptr3=new;  
 ptr3->coeff=ptr1->coeff;  
 ptr3->exp=ptr1->exp;  
 ptr3->link=NULL;  
 ptr1=ptr1->link;  
 }else{  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr3->link=new;  
 ptr3=new;  
 ptr3->coeff=ptr2->coeff;  
 ptr3->exp=ptr2->exp;  
 ptr3->link=NULL;  
 ptr2=ptr2->link;  
 }  
 }  
 if(ptr1!=NULL && ptr2==NULL){  
 while(ptr1!=NULL){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr3->link=new;  
 ptr3=new;  
 ptr3->coeff=ptr1->coeff;  
 ptr3->exp=ptr1->exp;  
 ptr3->link=NULL;  
 ptr1=ptr1->link;  
 }  
 }  
 if(ptr1==NULL && ptr2!=NULL){  
 while(ptr2!=NULL){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr3->link=new;  
 ptr3=new;  
 ptr3->coeff=ptr2->coeff;  
 ptr3->exp=ptr2->exp;  
 ptr3->link=NULL;  
 ptr2=ptr2->link;  
 }  
 }  
}  
void product(struct node\* header1,struct node\* header2,struct node\* header4){  
 struct node\* ptr1=header1->link;  
 struct node\* ptr2=header2->link;  
 struct node\* ptr4=header4;  
 if(ptr1==NULL && ptr2!=NULL){  
 while(ptr2!=NULL){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr4->link=new;  
 ptr4=new;  
 ptr4->coeff=ptr2->coeff;  
 ptr4->exp=ptr2->exp;  
 ptr4->link=NULL;  
 ptr2=ptr2->link;  
 }  
 }else if(ptr1!=NULL && ptr2==NULL){  
 while(ptr1!=NULL){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr4->link=new;  
 ptr4=new;  
 ptr4->coeff=ptr1->coeff;  
 ptr4->exp=ptr1->exp;  
 ptr4->link=NULL;  
 ptr1=ptr1->link;  
 }  
 }else{  
 while(ptr1!=NULL){  
 ptr2=header2->link;  
 while(ptr2!=NULL){  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr4->link=new;  
 ptr4=new;  
 ptr4->coeff=ptr1->coeff\*ptr2->coeff;  
 ptr4->exp=ptr1->exp+ptr2->exp;  
 ptr4->link=NULL;  
 ptr2=ptr2->link;  
 }  
 ptr1=ptr1->link;  
 }  
 }  
}  
void sort(struct node\* header,struct node\* header5){  
 struct node\* ptr=header->link;  
 struct node\* ptr1=header->link;  
 struct node\* ptr5=header5;  
 int exp,coeff;  
 if(ptr==NULL){  
 }else if(ptr->link==NULL){  
 }else{  
 while(ptr1->link!=NULL){  
 ptr=header->link;  
 while(ptr->link!=NULL){  
 if(ptr->exp < ptr->link->exp){  
 exp=ptr->exp;  
 coeff=ptr->coeff;  
 ptr->exp=ptr->link->exp;  
 ptr->coeff=ptr->link->coeff;  
 ptr->link->exp=exp;  
 ptr->link->coeff=coeff;  
 }  
 ptr=ptr->link;  
 }  
 ptr1=ptr1->link;  
 }  
 }  
 ptr=header->link;  
 while(ptr!=NULL){  
 if(ptr5->exp==ptr->exp){  
 ptr5->coeff=ptr5->coeff+ptr->coeff;  
 ptr5->exp=ptr->exp;  
 ptr=ptr->link;  
 }else{  
 struct node\* new = (struct node\*)malloc(sizeof(struct node));  
 ptr5->link=new;  
 ptr5=new;  
 ptr5->coeff=ptr->coeff;  
 ptr5->exp=ptr->exp;  
 ptr5->link=NULL;  
 ptr=ptr->link;  
 }  
 }  
  
}  
void display(struct node\* header){  
 struct node\* ptr=header;  
 while(ptr->link!=NULL){  
 ptr=ptr->link;  
 printf("(%dx^%d)",ptr->coeff,ptr->exp);  
 if(ptr->link!=NULL){  
 printf("+");  
 }  
 }  
}  
  
void main(){  
 int n1,n2,i;  
 struct node\* header1 = (struct node\*)malloc(sizeof(struct node));  
 header1->link=NULL;  
 struct node\* header2 = (struct node\*)malloc(sizeof(struct node));  
 header2->link=NULL;  
 struct node\* header3 = (struct node\*)malloc(sizeof(struct node));  
 header3->link=NULL;  
 struct node\* header4 = (struct node\*)malloc(sizeof(struct node));  
 header4->link=NULL;  
 struct node\* header5 = (struct node\*)malloc(sizeof(struct node));  
 header5->link=NULL;  
 printf("\nEnter the size of Polynomial-1 : ");  
 scanf("%d",&n1);  
 printf("\nEnter the Polynomial-1 \n");  
 for(i=0;i<n1;i++){  
 insert(header1);  
 }  
 printf("\nEnter the size of Polynomial-2 : ");  
 scanf("%d",&n2);  
 printf("\nEnter the Polynomial-2 \n");  
 for(i=0;i<n2;i++){  
 insert(header2);  
 }  
 sum(header1,header2,header3);  
 product(header1,header2,header4);  
 printf("\n Polynomial-1 : ");display(header1);  
 printf("\n Polynomial-2 : ");display(header2);  
 printf("\n Sum : ");display(header3);  
 printf("\n Product : ");sort(header4,header5);display(header5);  
}

**OUTPUT:**





**RESULT:**

Two polynomials are stored using linked list. Both are displayed along with their sum

and product.