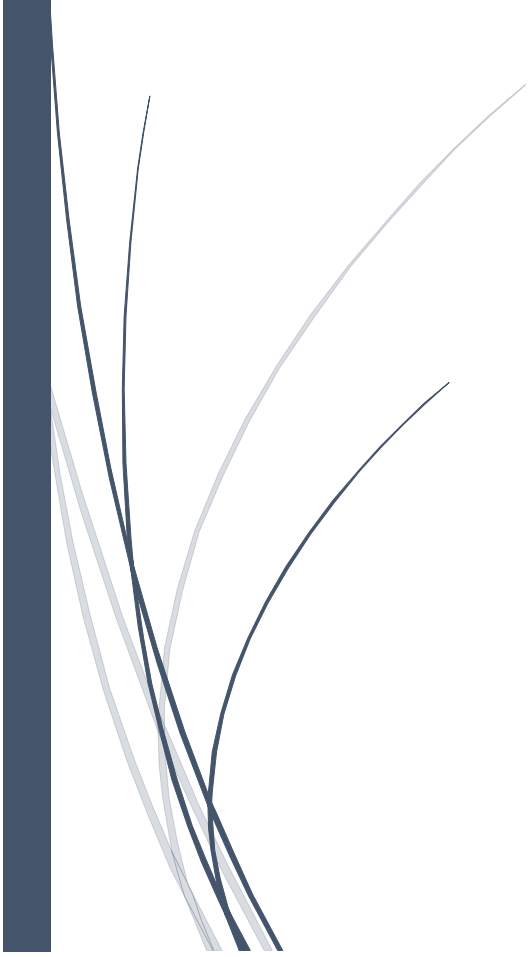
5/18/2022

Data Structure Project

Prof.Fadl Ba-Alwi



DHIYEA HUSSIEN AL THAIFANI

Data structure to solve the operation of Addicting , Subtraction and Multiplication of two Polynomials.

ABSTRACT DATA TYPE (ADT):

```
struct sll
{
float cofe;
int expo;
sll *link;
};
```

1) **GetNode();** to allocated momray for the node
2) **Enter ();** enter data for user
3) **InsertPoly(); to insert data and sort them in order**
4) **AddPoly(); to Add two** Polynomials
5) **SubtractPoly(); to** Subtraction two **polynomials**
6) **MultipliesPoly();to** Multiplication two **polynomials**
7) **Display();to print the list data**
8) **Search();to search for a item data in the polynomials**
9) **Sorting();Sorting after multiplies the polynomials if there is a like terms**
10) **Delete();For deleting items data int polynomials**

Algorithm of create and enter of the polynomial

Step1:

ask the user how many term he want to enter call **enter() function,**

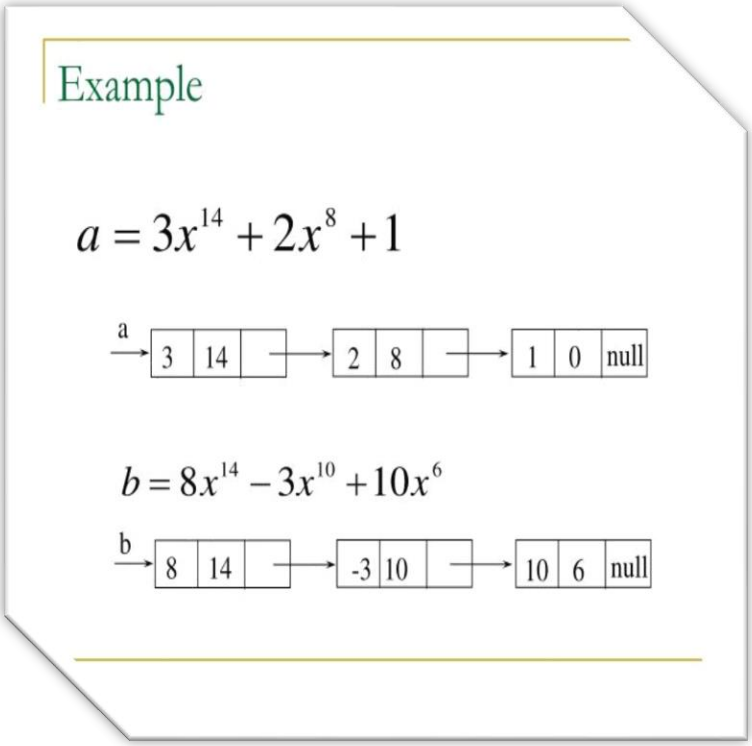
repeat step 2.

Step2:

than enter the terms data and call the **insert() function** and pass the **coefficients** with the **exponential** to the insert function that ordering the terms

And after enter each polynomial print it **display();**

And print the option that the user can chose from.

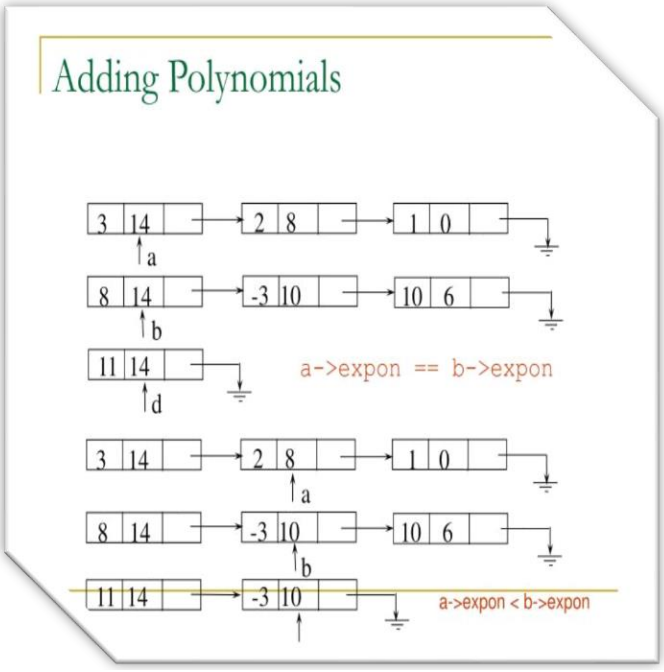


Algorithm to add and Subtracts two polynomials using linked list:

Step1: Let **Poly1($5x^3 + 7x^2$)** and

poly2($5x^2 + 4x^1 + 2x^0$) be the two polynomials represented by linked lists 1,2**.**
while **Poly1** and **poly2** be **are not null,**
repeat step2.

Step2: If powers of the two terms ate equal then if the terms do not cancel **(add or subtract)** coefficients of **poly1** and **poly2** then insert the result of the terms into the **poly3** Polynomial
Advance poly1, Advance poly2
Poly1=poly1->link;
Poly2=poly->link;
Else if the power of the **first** polynomial **> power of second** Then insert the term from **first polynomial**
into **poly3** polynomial
Advance poly1 .Poly1=poly1->link;
Else insert the term from **second** polynomial into **poly3** polynomial
Advance poly2



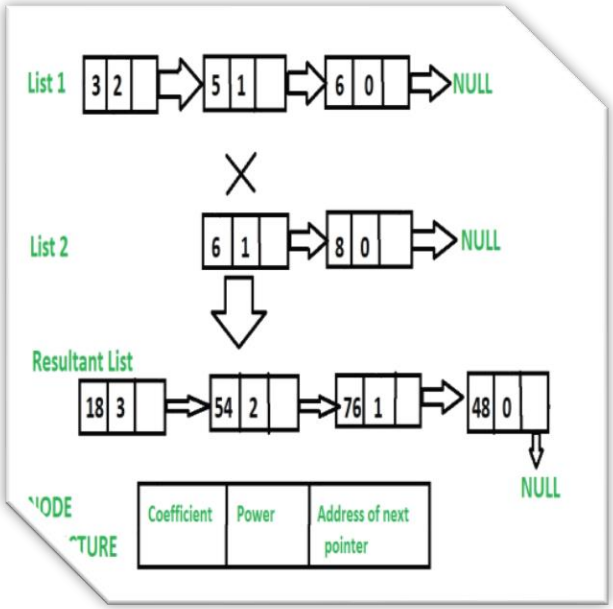
Step2: copy the remaining terms from the non empty polynomial into the **poly3** polynomial**.**
The third step of the algorithm is to be processed till the end of the polynomials has not been reached.

Algorithm of Multiplication of two polynomials using Linked List:

1) First, we will traverse the **Poly1** and multiply each of its nodes, with each and every node in **Poly2**.

- We will multiply the power of the nodes and store it in a variable **power**.
- We will multiply the coefficients of the nodes and store it in a variable **coeff**.
- Then we will add a new node in **Poly3** with **power** and **coeff** values.
  2) We will perform **step 1** until each node in **Poly1** has been multiplied with every node of **Poly2**.
  3) After all the terms in the given polynomials are multiplied and stored in a new list **Poly3**, then we will combine the different terms with the same power.

- We will traverse the **Poly3** to compare the **power** of each factor, starting with the next node.
- If the **power** is same for any factors, we will add the coefficients and remove the other duplicate node.
  4) So now what we have is the required polynomial, and we can print the answer.



Algorithm for searching of term in the list:

Suppose poly is the address of the first node in the linked list and DATA is the information to be searched. After searching, if the DATA is found, POS will contain the corresponding position in the list.

1. Input the DATA to be searched

2. Initialize TEMP =poly; POS =1;

3. Repeat the step 4, 5 and 6 until (TEMP is equal to NULL)

4. If (TEMP → DATA is equal to DATA)

(a) Display "The data is found at POS"

(b) Exit

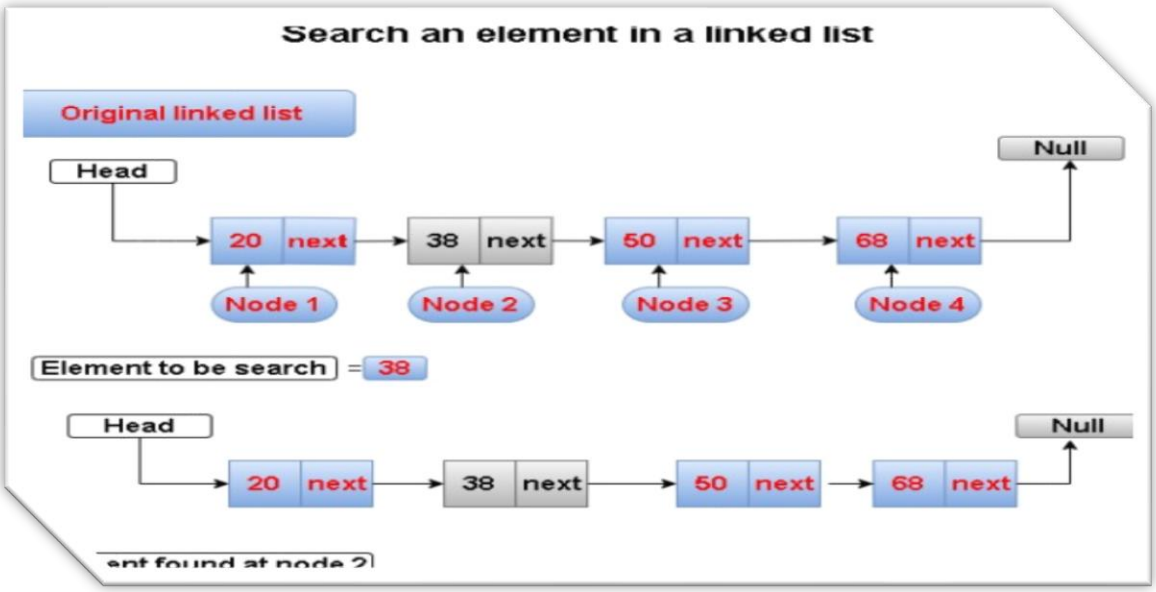
5. TEMP = TEMP → Next

6. POS = POS+1

7. If (TEMP is equal to NULL)

(a) Display "The data is not found in the list"

8. Exit



Algorithm for deleting term of the list:

uppose poly is the first position in linked list. Let DATA be the element to be deleted. TEMP, HOLD is a temporary pointer to hold the node address.

1. Input the DATA to be deleted

2. if ((START → DATA) is equal to DATA)

(a) TEMP = START

(b) START = START → Next

(c) Set free the node TEMP, which is deleted

(d) Exit

. HOLD = START

4. while ((HOLD → Next → Next) not equal to NULL))

(a) if ((HOLD → NEXT → DATA) equal to DATA)

(i) TEMP = HOLD → Next

(ii) HOLD → Next = TEMP → Next

(iii) Set free the node TEMP, which is deleted

(iv) Exit

(b) HOLD = HOLD → Next

5. if ((HOLD → next → DATA) == DATA)

(a) TEMP = HOLD → Next

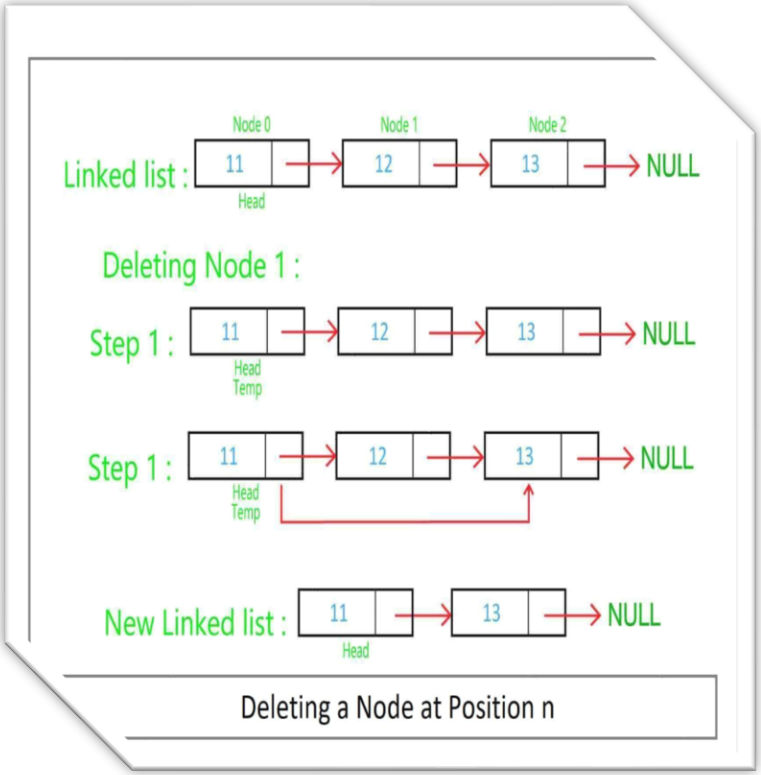
(b) Set free the node TEMP, which is deleted

(c) HOLD → Next = NULL

(d) Exit

6. Disply "DATA not found"

7. Exit



Algorithm for display data of the list:

uppose Poly is the address of the first node in the linked list. Following algorithm will visit all nodes from the Poly node to the end.

1. If (START is equal to NULL)

(a) Display “The list is Empty”

(b) Exit

2. Initialize TEMP = Poly

3. Repeat the step 4 and 5 until (TEMP == NULL)

4. Display “TEMP → DATA”

5. TEMP = TEMP → Next

6. Exit

Output:

##

this program is a Data Structures implementing of the single linked list
Please enter the polynomial equations:

polynomial 1 :

How many terma you want to enter?

Enter the coefficient for the tarme

Enter the expo for the tarme

polynomial 2 :

How many terma you want to enter?

Enter the coefficient for the tarme

Enter the expo for the tarme

***********************"MAIN MENU *********************"**

1:For adding polynomial press number:

2:For subtract polynomial press number:

3:For multiplies polynomial press number:

4:For updating the polynomial press number:

5:For search about any term in the two Polynomails:

6:For delete any term in the two Polynomails:

7:For exite press:

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct sll
{

        float cofe;
        int expo;
        sll *link;
};
sll* getNode ()
{

        return((sll*)malloc(sizeof(sll)));
}
sll* insertPoly (sll*start ,int ex , float cof)

{
        sll* temp;
        sll*p;
        temp=getNode();
        temp->cofe=cof;
        temp->expo=ex;
        if (start==NULL || ex>start->expo)
        {//insert in the first //
                temp->link=start;
                start=temp;}
        else
        {//insert in end //
                p=start;
                while (p->link != NULL &&p->link->expo>ex)

                        p=p->link;
                        temp->link=p->link;
                        p->link=temp;
                        if (p->link==NULL)
                        temp->link=NULL;    }
        return start;


}//End of insert //
```

```
sll* enter(sll*stert)
{
	int n,expo;
	float cof;



	cout<<"How many terma you want to enter:";
	cin>>n;
	cout<<"                                  \t \t# \n";
	for(int i=0;i<n;i++)
	{
		cout<<"Enter the cofntion for the tarme "<<i+1<<":";
		cin>>cof;
		cout<<"                                  \t \t# \n";
		cout<<"Enter the expo for the tarme "<<i+1<<":";
		cin>>expo;
		cout<<"                                  \t \t# \n";
		stert=insertPoly(stert , expo,cof);
	}
	return stert;
}
sll* addPoly(sll*p1,sll*p2)
{

	sll*ptr1=p1;
	sll*ptr2=p2;
	sll*p3_start=NULL;

	while(ptr1!=NULL && ptr2!=NULL)
	{
		if (ptr1->expo==ptr2->expo)
		{
		//polynomial creat and stor the result of adding two polynomials
	p3_start= insertPoly(p3_start,ptr1->expo ,ptr1->cofe + ptr2->cofe);//or p2->expo
			ptr1=ptr1->link;
			ptr2=ptr2->link;
		}
		else if(ptr1->expo > ptr2->expo)
		{
			p3_start=insertPoly(p3_start,ptr1->expo,ptr1->cofe);
			ptr1=ptr1->link;
		}
		else if(ptr2->expo > ptr1->expo)
		{
			p3_start=insertPoly(p3_start,ptr2->expo,ptr2->cofe);
			ptr2=ptr2->link;
		}
	}
	while(ptr1!=NULL)
```

```
	{
		p3_start=insertPoly(p3_start,ptr1->expo,ptr1->cofe);
		ptr1=ptr1->link;
	}
		while(ptr2!=NULL)
	{
		p3_start=insertPoly(p3_start,ptr2->expo,ptr2->cofe);
		ptr2=ptr2->link;
	}
	return p3_start;
}

sll* subtractPolt(sll*p1,sll*p2)
{

	sll*ptr1=p1;
	sll*ptr2=p2;
	sll*p3_start=NULL;

	while(ptr1!=NULL && ptr2!=NULL)
	{
		if (ptr1->expo==ptr2->expo)
		{
			//polynomial creat and stor the tesult of subtract two polynomials
			p3_start= insertPoly(p3_start,ptr1->expo ,ptr1->cofe - ptr2-
>cofe);//or p2->expo
			ptr1=ptr1->link;
			ptr2=ptr2->link;
		}
		else if(ptr1->expo > ptr2->expo)
		{
			p3_start=insertPoly(p3_start,ptr1->expo,ptr1->cofe);
			ptr1=ptr1->link;
		}
		else if(ptr2->expo > ptr1->expo)
		{
			p3_start=insertPoly(p3_start,ptr2->expo,ptr2->cofe);
			ptr2=ptr2->link;
		}
	}
	while(ptr1!=NULL)
	{
		p3_start=insertPoly(p3_start,ptr1->expo,ptr1->cofe);
		ptr1=ptr1->link;
	}
		while(ptr2!=NULL)
	{
		p3_start=insertPoly(p3_start,ptr2->expo,ptr2->cofe);
		ptr2=ptr2->link;
	}


	return p3_start;
```

```
}

void display(sll*p)
{
        sll*tmep;
        //check if the list is empty and print that
        if(p==NULL)
        {
        cout<<"SORRY the list is empty\n";
        return ;
        }
        //print the conf and expo of the nodes one by one
         tmep=p;
        while (tmep!=NULL)
        {
                cout<<"("<<tmep->cofe<<"x^"<<tmep->expo<<")";
                tmep=tmep->link;
        if(tmep!=NULL)
                cout<<"+";
                else
                cout<<"\n";
        }
        cout<<endl;
}


//founctoin to multiples
sll* Multiplies (sll*p1,sll*p2)
{
        sll*ptr1=p1,*ptr2=p2;
        sll*p3_start=NULL;

        while (ptr1!=NULL)
        {
                ptr2=p2;
                while(ptr2!=NULL)
                {

                        p3_start=insertPoly(p3_start,ptr1->expo+ptr2->expo,ptr1-
>cofe*ptr2->cofe);
                        ptr2=ptr2->link;
                }

                ptr1=ptr1->link;
        }
        //return p3_start;
        display(p3_start);

        return p3_start ;}
void search(sll*p1,sll*p2)
{
        sll*ptr=p1,*ptr2=p2;
        sll*term=NULL;
```

```
	int ex;
	float cof;
	cout<<"enter the the term that want to find\n";
	cout<<"enter the cofntion of that term:";
	cin>>cof;
	cout<<"enter the expontion of that term:";
	cin>>ex;
	term=insertPoly(term,ex,cof);

	//ptr=p1;

	while(ptr!=NULL||ptr2!=NULL)
	{
		if ((ptr->cofe==term->cofe&&ptr->expo==term->expo)|| (ptr2-
>cofe==term->cofe&&ptr2->expo==term->expo))
		{
			cout<<"the term is fonde in the first polynomail.\n";
			return;
		}
		ptr=ptr->link;


	}
	if (ptr==NULL ||ptr2==NULL)
	cout<<"the term is not fonde in the secande polynomail.\n";
}
void storing(sll*p3)
{
		sll*ptr3=p3,*tmp=NULL;
	//ptr3=p3_start;
	//tmp=NULL;
	while(ptr3->link!=NULL)
	{
		if(ptr3->expo==ptr3->link->expo)
		{
			ptr3->cofe=ptr3->cofe+ptr3->link->cofe;
			tmp=ptr3->link;

			ptr3=ptr3->link->link;
			free(tmp);
			//tmp=NULL;
			//return;
		}
		else{
			ptr3=ptr3->link;
		}

	}
	display(p3);}
void propet()
{
		cout<<"**********WELCOME*************\n \n";
	/*cout<<"////////////////////////\n";
```

```
        cout<<"//DS for soleving ///////\n";
           cout<< "//polynomial equation////////\n";
             cout<<"// using single linked liste///\n";
             cout<<"////////////////////////////\n";*/

           cout<<"############################################################\n";
           cout<<"this program is a Data Structures empentions of the single linked list\n";
           cout<<"Please enter the polynomial equaotoins ";
}
void Delete (sll*p1,sll*p2)
{
           int expo;
           float cof;
           sll*ptr1=p1,*ptr2=p2,*temp;
           cout<<"enter the data of the term that you want to delete\n";
           cout<<"enter cofetion:";
           cin>>cof;
           cout<<"enter expontion:";
           cin>>expo;
           if(ptr1->cofe==cof&&ptr1->expo==expo)
           {
                      temp=p1;
                      p1=temp->link;
                      free(temp);/*First element deleted in the first polynomail */
                      cout<<"term deleted from the first polynomail";
                      return;

           }
                      if(ptr2->cofe==cof&&ptr2->expo==expo)
           {
                      temp=p2;
                      p2=temp->link;
                      free(temp);/*First element deleted in the second polynomail*/
                      cout<<"term deleted from the second polynomail";
                      return;

           }
           ptr1=p1;
           while(ptr1->link->link!=NULL||ptr2->link->link!=NULL)
           {
                      if(ptr1->link->cofe==cof&&ptr1->link->expo==expo)/*Element deleted in
between*/
                      {
                                 temp=ptr1->link;
                                 ptr1->link=temp->link;
                                 free(temp);
                                 cout<<"term deleted from the first polynomail\n";
                                            return;
                      }
                      else if(ptr2->link->cofe==cof&&ptr2->link->expo==expo)
                      {
                                 temp=ptr2->link;
                                 ptr2->link=temp->link;
```

```
                        free(temp);
                        cout<<"term deleted from the second polynomail\n";
                                return;
                }
                ptr1=ptr1->link;
                ptr2=ptr2->link;

        }/*End of while */
         /*Last element deleted*/
        if(ptr1->link->cofe==cof&ptr1->link->expo==expo)
        {
                temp=ptr1->link;
                free(temp);
                ptr1->link=NULL;
                return;
        }
    else if(ptr2->link->cofe==cof&ptr2->link->expo==expo)
        {
                temp=ptr2->link;
                free(temp);
                ptr2->link=NULL;
                cout<<"term deleted from the second polynomail\n";
                return;
        }
        else
        cout<<"the data of the term are not found\n";
}

int main ()
{
        sll*pol1=NULL;
        sll*pol2=NULL;
        sll*pol3=NULL;
        int flage;
        propet();
      cout<<" polynomial 1\n";
           pol1=enter(pol1);
           display(pol1);
        cout<<" polynomial 2\n";
        pol2=enter(pol2);
      display(pol2);
        do
        {
                cout<<"1-For adding polynomial press number:1  \t \t# \n";
                cout<<"2-For subtract polynomial press number:2 \t \t#\n";
                cout<<"3-For multiplies polynomial press number:3 \t \t# \n";
                cout<<"4-For updating the polynomial press number:4 \t \t# \n";
                cout<<"5-For search about any term in the two Polynomails:5 \t \t# \n";
                cout<<"6-For delete  any term in the two Polynomails:6 \t \t# \n";
                cout<<"7-For exite press:7                      \t \t#\n";
                cin>>flage;
                cout<<"                                 \t \t# \n";
```

```
        switch(flage)
        {
                case 1:
                 pol3=addPoly(pol1,pol2);
            cout<<"The result is\n";
                 display(pol3);
                 break;
                 case 2:
                 pol3= subtractPolt(pol1,pol2);
                 cout<<"The result is\n";
                 display(pol3);
                 break;
                 case 3:
                pol3= Multiplies(pol1,pol2);
                cout<<"The result is\n";
                 storing(pol3);
                 break;
                  case 4:
            cout<<" polynomial 1\n";
            pol1=enter(pol1);
            display(pol1);
         cout<<" polynomial 2\n";
      pol2=enter(pol2);
      break;
       case 5:
      search(pol1,pol2);
      break;
      case 6:
      Delete(pol1,pol2);
      break;
        case 7:
      cout<<"bye.....^_^\n";
      cout<<"##################################################\n";
      break;
      default:
      cout<<"wrong potion\n";
      break;
         }
         }

         while(flage!=7);
}
```