U10P32002 Spring 2021

ABID ALI, 2019380141

Abiduu354@gmail.com, (04/0/2021)

**◎ Problem Description**

In this assignment we are going to write a C++program file (“dse\_assign02.cpp”and text file also attached) which includes implementations of the methods of “Single-variable Polynomial Addition and Multiplication”. These code were in CodeBlocks C++(IDE). The purpose of the program is that user can identify the single-variable polynomial’s addition and multiplication. The user can then visualize the processes by this program.

**◎ Goals**

This assignment is designed to help understanding and mastering the use of C++ header files, basic C++ syntax, and stream-based I/O. By successfully completing this assignment we could master the following outcomes:

* **Understanding the of basic C++ syntax.**
* **By using basic C++ we learn implementation of control structures.**
* **We would learn add and multiply two polynomials**.
* **We need the output polynomial to be sorted by exponent and has at most one term of any power.**
* **Time complexity of multiplying polynomial is O(n2)**
* **ime complexity of addition polynomial is O(m + n)**

**◎ Requirement Analysis**

Input:

* First line – the total number of terms in the first polynomial n, the coefficient and exponent of each term in ascending order;
* Second line -- the total number of terms in the second polynomial m, the coefficient and exponent of each term in ascending order.

Output:

* First line – the coefficient and exponent of each term in the sum in ascending order;
* Second line -- the coefficient and exponent of each term in the multiplication in ascend order

**Sample input:**

4 1 0 -10 6 2 8 7 14 []

5 -1 4 10 6 -3 10 8 14 4 18 []

Sample output:

1 0 -1 4 2 8 -3 10 15 14 4 18 []

-1 4 10 6 7 10 -102 12 28 14 30 16 -9 18 -10 20 16 22 -61 24 8 26 56 28 28 32

[]

Sample input:

4 -2 0 6 1 -5 2 3 4

3 3 1 -7 4 5 20

Sample output:

-2 0 9 1 -5 2 -4 4 5 20

-6 1 18 2 -15 3 14 4 -33 5 35 6 -21 8 -10 20 30 21 -25 22 15 24

**◎ General Design**

**Data Structure**

* **Polynomials** are algebraic expressions that consist of variables and coefficients. Variables are also sometimes called indeterminates. We can perform [arithmetic operations](https://byjus.com/maths/arithmetic-operations/) such as addition, subtraction, multiplication and also positive integer exponents for polynomial expressions but not division by variable.
* A polynomial can be written as the sum of a finite number of terms. The product of a constant(called the coefficient of the term)and a finite number of variables(usually represented by letters)raised to integer powers.

KEY TERMINOLOGY:

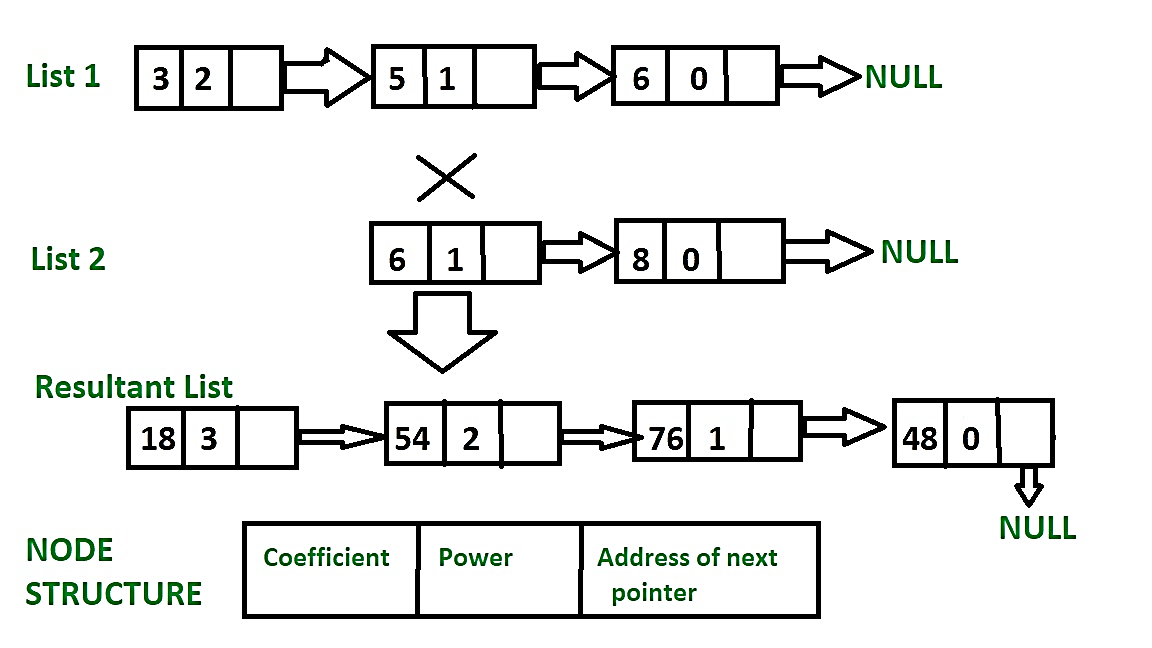
* **Polynomial**: An expression consisting of a sum of a finite number of terms, each term being the product of a constant coefficient and one or more variables raised to a non-negative integer power
* **Degree**: The sum of the exponents of a term; the order of a polynomial.
* **Coefficient**: A constant by which an algebraic term is multiplied.

**The basic operation(addition )of algorithm:**

* The rules for adding and subtracting algebraic expressions apply to polynomials; only like terms can be combined.
* Any two polynomials can be added or subtracted, regardless of the number of terms in each, or the degrees of the polynomials.
* The sum or difference of two polynomials will have the same degree as the polynomial with the higher degree in the problem.

**The basic operation(multiplication) of algorithm:**

* In this approach we will multiply the 2nd polynomial with each term of 1st polynomial.
* Store the multiplied value in a new linked list.
* Then we will add the coefficients of elements having the same power in resultant polynomial.



**◎ Implementation**

A polynomial p(x) is the expression in variable x which is in the form (axn + bxn-1 + …. + jx+ k), where a, b, c …., k fall in the category of real numbers and 'n' is non negative integer, which is called the degree of polynomial.

An essential characteristic of the polynomial is that each term in the polynomial expression consists of two parts:

* one is the coefficient
* other is the exponent

Example:

10x2+ 26x, here 10 and 26 are coefficients and 2, 1 is its exponential value.

Points to keep in Mind while working with Polynomials:

* The sign of each coefficient and exponent is stored within the coefficient and the exponent itself
* Additional terms having equal exponent is possible one
* The storage allocation for each term in the polynomial must be done in ascending and descending order of their exponent

Polynomial can be represented in the various ways. These are:

* By the use of arrays
* By the use of Linked List

**The use of Linked List:**

**According to teacher’s instruction student info. was given**

/\*

\*This code was done in C++

\*Merge Two Sorted Linked Lists

\*Assignment 2

\*ABID ALI\_2019380141

\*abiduu354@gmail.com

\*/

**Application of program:** The beginning of the program which is contains header files.

#include<conio.h>

#include<stdlib.h>

#include<bits/stdc++.h>

using namespace std;

//Declaring poly data type

typedef struct node

{

int coef;/\* coefficient \*/

int expo;/\* exponent \*/

struct node \*link;

}node;

**Insertion:**

1)We build and declare our function here.

2)We will use these function when we need to do different operation.Such as:Traverse,add,multiply

//We are creating instances such as insert,traverse, use , add and multiply

void insert\_value(struct node \*\*,int,int);

void traverse(struct node \*);

void poly\_add(struct node \*\*,struct node \*\*,struct node \*\*);

void poly\_multi(struct node \*\*,struct node \*\*,struct node \*\*);

int main()

{

struct node \*start\_p=NULL,\*start\_q=NULL,\*start\_r=NULL;

int i,n,c,e;

cin>>n;

for(i=0; i<n; i++)

{

cin>>c;

cin>>e;

insert\_value(&start\_p,c,e);

}

cin>>n;

for(i=0; i<n; i++)

{

cin>>c;

cin>>e;

insert\_value(&start\_q,c,e);

}

poly\_add(&start\_p,&start\_q,&start\_r);

//Sum of two polynomial values were calculated in this function

traverse(start\_r);

start\_r=NULL;

poly\_multi(&start\_p,&start\_q,&start\_r);

//Product of the two polynomial values were calculated in this function

traverse(start\_r);

Structure–Inserting value:

1)A polynomial as oppose to the monomial is a sum of monomials where each monomial is called a term. The degree of the polynomial is the greatest degree of its terms. A polynomial is usually written with the term with the highest exponent of the variable first and then decreasing from left to right. The first term of a polynomial is called the leading coefficient.

2) To multiply two polynomials together, multiply every term of one polynomial by every term of the other polynomial.

3)The zeros of a product of two polynomial are the zeros of the two factors, combined together.

4) We use traverse function to chane the order and then it becomes in ascending order.Such as



}

void traverse(struct node \*start)

{

struct node \*current;

current = start;

if (current == NULL)

{

cout << current << "Empty polynomial number";

}

else

{

while (current != NULL)

{

if (current->coef == 0)

{

current = current->link;

continue;

}

cout << current->coef << " "<< current->expo << " ";

current = current->link;

if (current != NULL)

cout << " ";

else

cout << "\n";

}

}

}

void poly\_add(struct node \*\*start\_p, struct node \*\*start\_q, struct node \*\*start\_r)

{

int c, e;

struct node \*pptr, \*qptr;

\*start\_r = NULL;

pptr = \*start\_p;

qptr = \*start\_q;

while (pptr != NULL && qptr != NULL)

{

if (pptr->expo == qptr->expo)

{

c = pptr->coef + qptr->coef;

e = pptr->expo;

insert\_value(start\_r, c, e);

pptr = pptr->link;

qptr = qptr->link;

}

else

{

if (pptr->expo > qptr->expo)

{

c = pptr->coef;

e = pptr->expo;

insert\_value(start\_r, c, e);

pptr = pptr->link;

}

else

{

c = qptr->coef;

e = qptr->expo;

insert\_value(start\_r, c, e);

qptr = qptr->link;

}

}

}

Search:

Polynomials and rational functions are used for approximation in many everyday devices. For example, every time we take a picture with a smartphone, our phone looks at some data points and fills in the appropriate colors in the blanks, thus saving us a lot of memory, with the help of rational functions. Every time we say something through the phone, our phone tries to reduce the background noise by approximating our sound for short periods of time, again with the help of rational functions.

* They are well studied: a lot of their properties are known.
* They are easy to compute, using only multiplication and addition.
* They are closed under rescaling or changing of locations: if we change a kilometer to a mile we still get a polynomial.
* [Polynomial](https://oer2go.org/mods/en-boundless/www.boundless.com/algebra/definition/polynomial/index.html) [functions](https://oer2go.org/mods/en-boundless/www.boundless.com/algebra/definition/function/index.html) are easy to use for modeling, but not suitable to modeling [asymptotes](https://oer2go.org/mods/en-boundless/www.boundless.com/algebra/definition/asymptote/index.html) ,some functional forms and rational functions that are more difficult to use and can include undesirable asymptotes.

while (pptr != NULL)

{

c = pptr->coef;

e = pptr->expo;

insert\_value(start\_r, c, e);

pptr = pptr->link;

}

while (qptr != NULL)

{

c = qptr->coef;

e = qptr->expo;

insert\_value(start\_r, c, e);

qptr = qptr->link;

}

}

void poly\_multi(struct node \*\*start\_p, struct node \*\*start\_q, struct node \*\*start\_r)

{

int c, e;

struct node \*pptr, \*qptr;

\*start\_r = NULL;

pptr = \*start\_p;

qptr = \*start\_q;

if (\*start\_p == NULL && \*start\_q == NULL)

{

cout << start\_p << start\_q << "Multiplication of polynomial is not possible!";//Warning is given for invalid value.

}

else

{

while (pptr != NULL)

{

qptr = \*start\_q;

while (qptr != NULL)

{

c = pptr->coef \* qptr->coef;

e = pptr->expo + qptr->expo;

insert\_value(start\_r, c, e);

qptr = qptr->link;

}

pptr = pptr->link;

}

}

}

void insert\_value(struct node \*\*start,int c,int e)

{

struct node \*current,\*current1,\*prev;

if (\*start==NULL)

{

current=(struct node\*)malloc(sizeof(struct node));

if (current==NULL)

cout << current << "Node is cann't be created, value cann't be inserted";

else

{

current->coef=c;

current->expo=e;

current->link=NULL;

\*start=current;

}

}

else

{

current1=\*start;

//Inserting value in the ascending order

while (current1!=NULL && current1->expo<e)

{

prev=current1;

current1=current1->link;

}

if(current1==NULL)

{

current=(struct node \*)malloc(sizeof(struct node));

if(current==NULL)

{

cout << current << "Node is not found";

}

else

{

current->coef=c;

current->expo=e;

current->link=NULL;

prev->link=current;

}

}

else

{

if(current1->expo==e)

current1->coef=current1->coef+c;

else

{

if(current1==\*start)

{

current=(struct node \*)malloc(sizeof (struct node));

if(current==NULL)

{

cout << current << "Node is not created";

}

else

{

current->coef=c;

current->expo=e;

current->link=\*start;

\*start=current;

}

}

else

{

current=(struct node \*)malloc(sizeof(struct node));

if (current==NULL)

{

cout << current << "Node is not created";

}

else

{

current->coef=c;

current->expo=e;

current->link=current1;

prev->link=current;

}

}

}

}

}

}

**Display − Displays the complete list:**

1)A polynomial is written in descending order if its terms are arranged in order from largest degree to smallest degree. Note that this is the standard form for a polynomial; rearranging your polynomials to descending order should be automatic.

2)We use traverse function to chane the order and then it becomes in ascending order.Such as



void traverse(struct node \*start)

{

    struct node \*temp;

    temp = start;

    if (temp == NULL)

    {

        Cin>>"Empty polynomial">>end;l

    }

    else

    {

        while (temp != NULL)

        {

            if (temp->coefficient == 0)

            {

                temp = temp->link;

                continue;

            }

            printf("%d %d", temp->coefficient, temp->exponent);

            temp = temp->link;

            if (temp != NULL)

                cout<<" "<<;

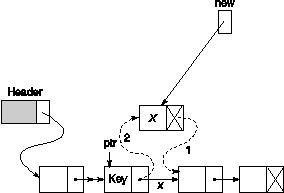
            else

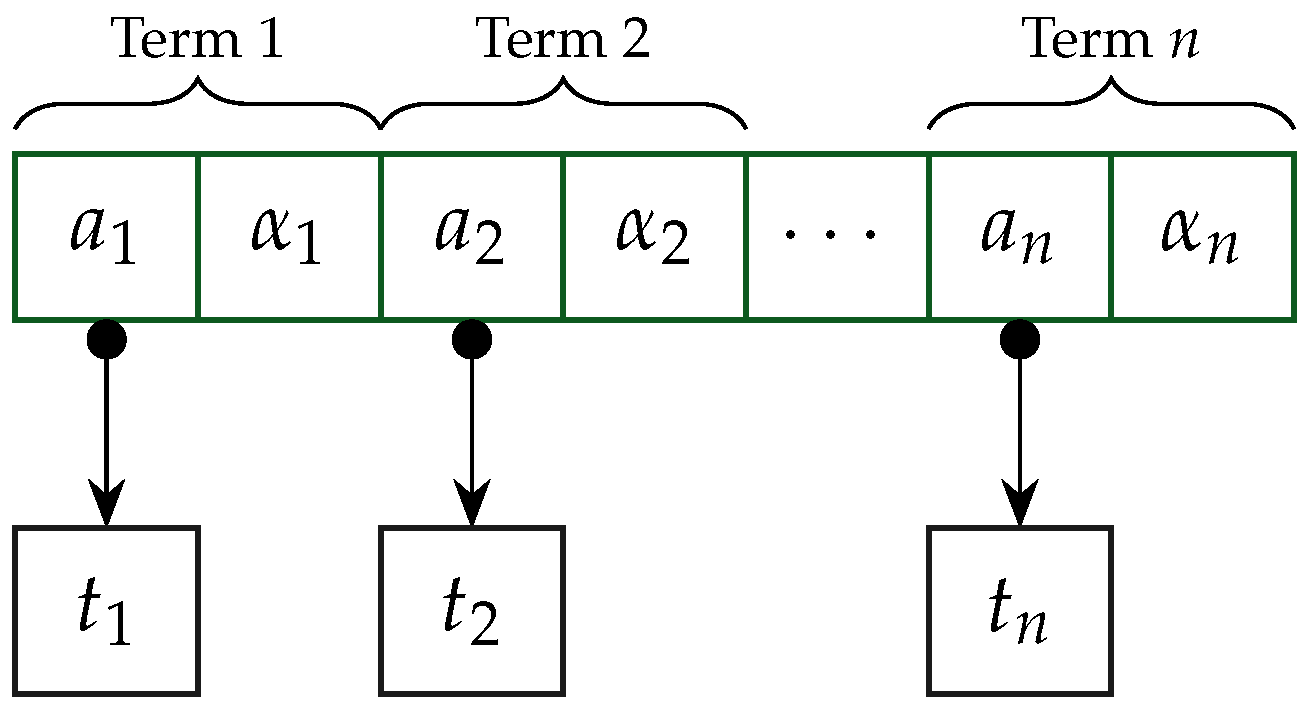
                cout<<" "<<endl;

        }

    }

}





**◎ Test Description and Results**

**Test 1:**

**Input:**

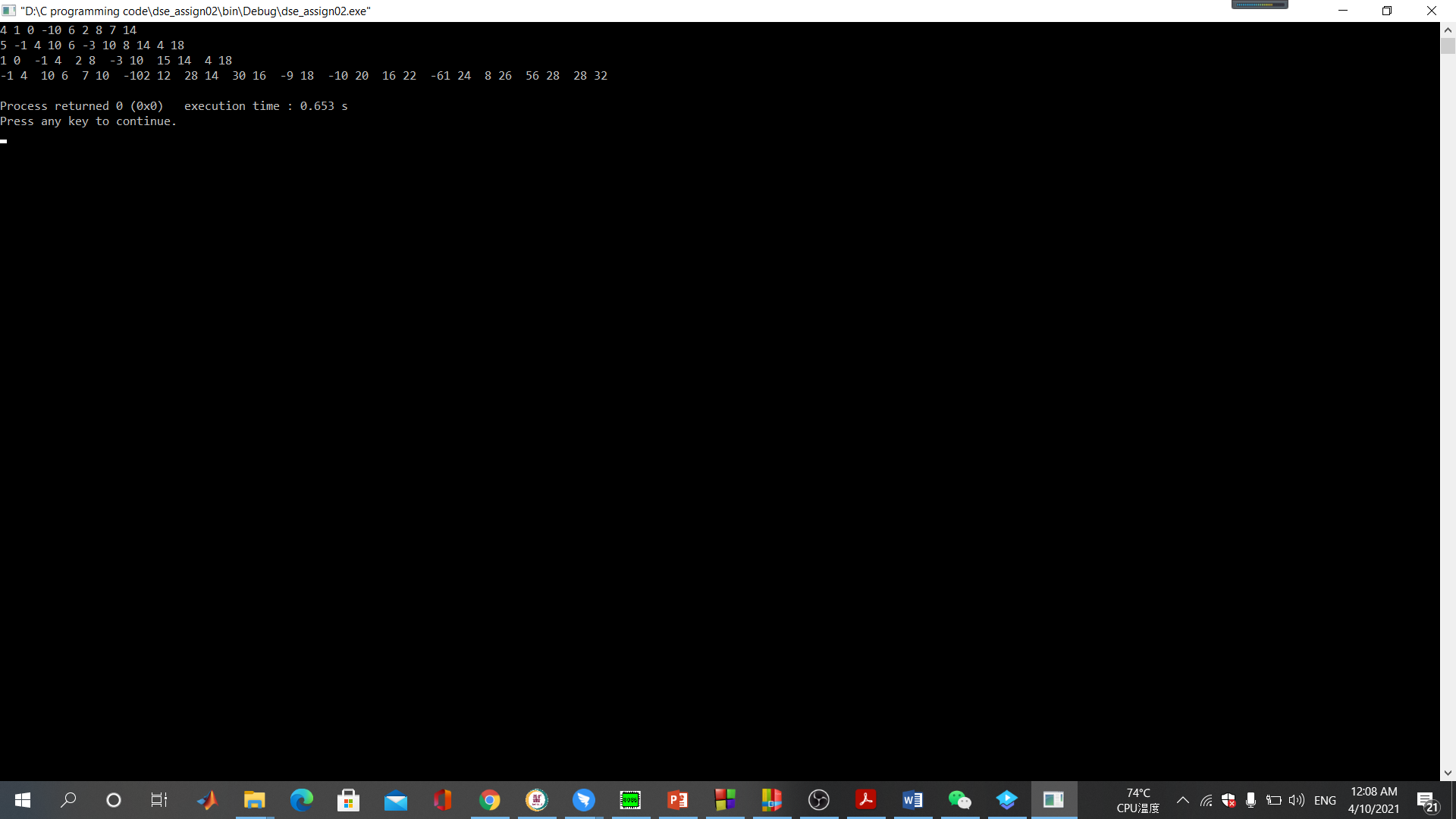
4 1 0 -10 6 2 8 7 14

5 -1 4 10 6 -3 10 8 14 4 18

**Output:**

1 0 -1 4 2 8 -3 10 15 14 4 18

-1 4 10 6 7 10 -102 12 28 14 30 16 -9 18 -10 20 16 22 -61 24 8 26 56 28 28 32



**Picture: Test 1**

**Test 2:**

**Input:**

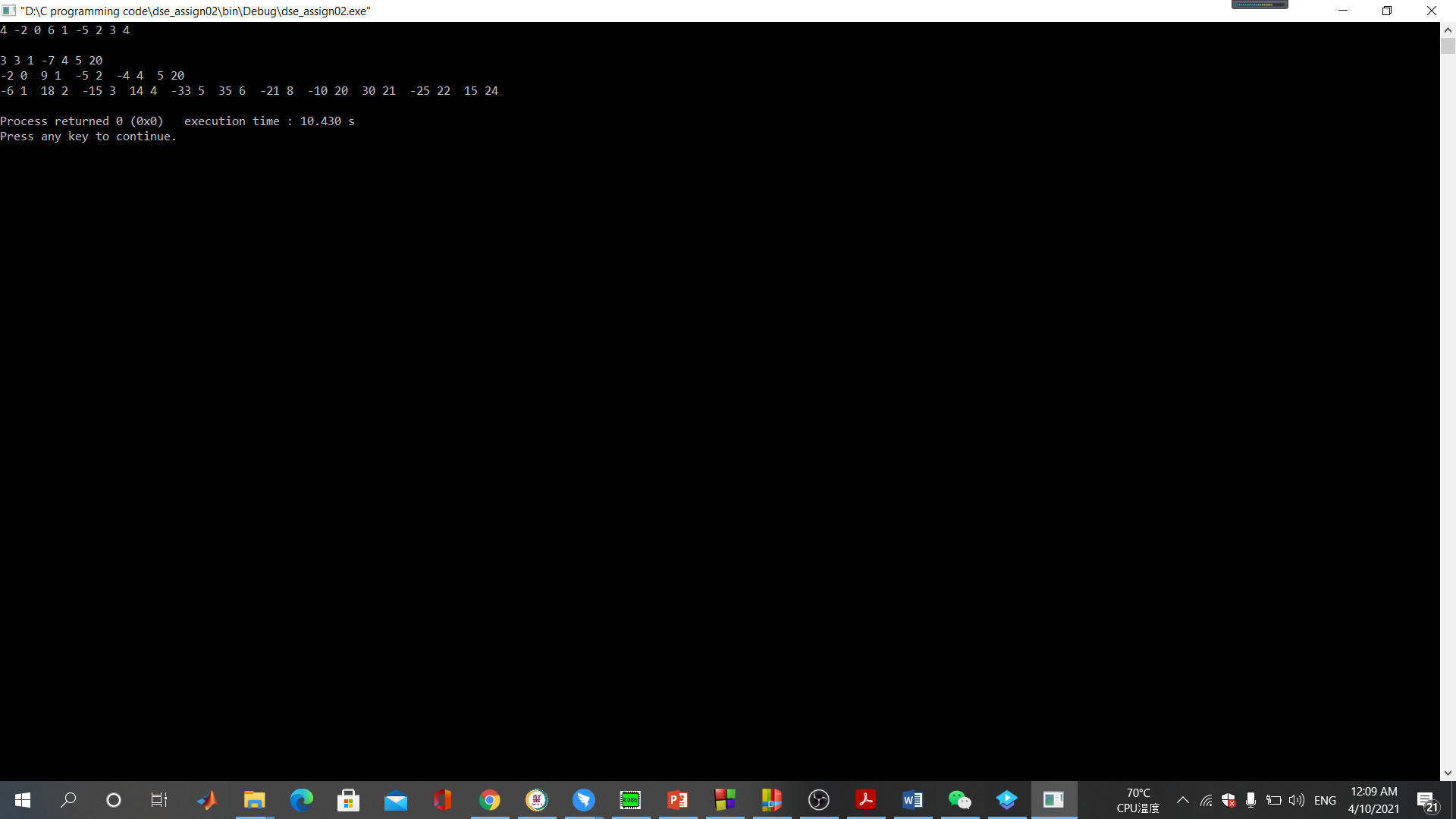
4 -2 0 6 1 -5 2 3 4

3 3 1 -7 4 5 20

**Output:**

-2 0 9 1 -5 2 -4 4 5 20

-6 1 18 2 -15 3 14 4 -33 5 35 6 -21 8 -10 20 30 21 -25 22 15 24



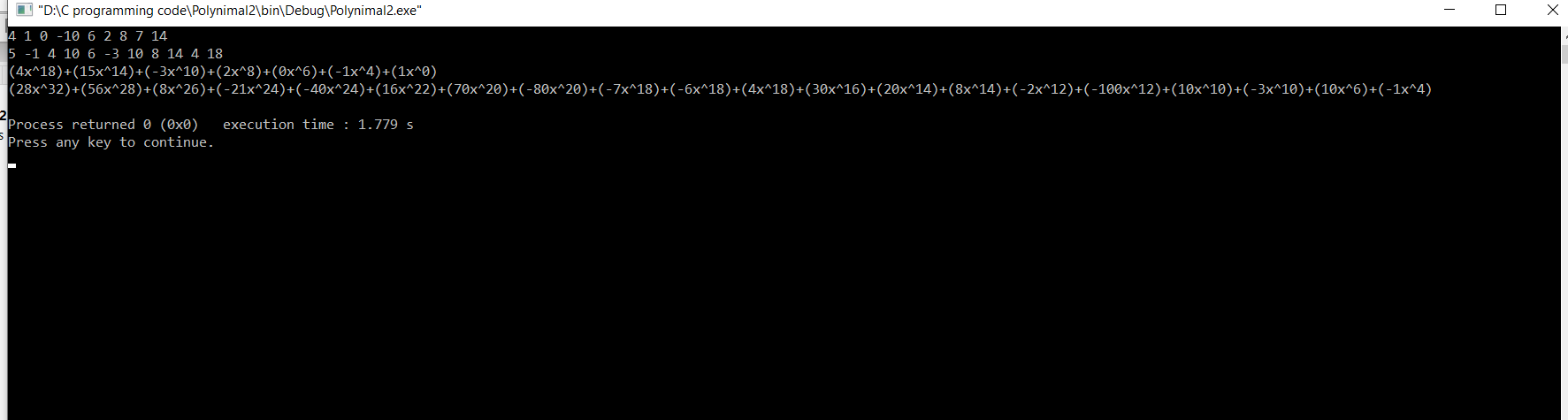
**Picture: Test 2**

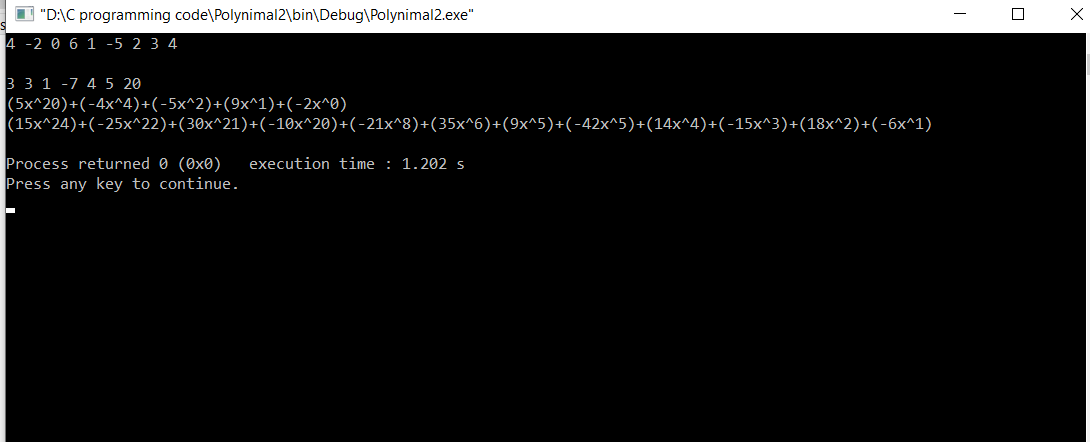
**Remarks:**

We know that, polynomial has “x” for better illustration.I made it look like a polynomial function.Looks very elegance along with logical.

**BUG:**

There were many problems in the program at the beginning for example polynomial has “x” then I started to debug the program step by step and try to understand the concept and fixed some function.The bug was reduce massively.It shows some minor error in output. Then,I realize ,I didn’t follow the order properly.Because,I was confused with one part.After discussing with my classmates ,they point out my mistakes.Then,I mixed the code using their suggestion.My multiplication value was creating problem because I used few wrong function and my value were not being stored.

****

****

All those errors were fixed and I am resubmitting the problem again.

**◎ Epilogue**

While doing this assignment I had many bugs which had influence on program working and many troubles with compiling the program, because of syntax and doing arithmetic calculation and turning it polynomial form was difficult at first.

Finally after discussing with my classsmates they give me a idea .Then,I realize the mistake that I was making.After along time,I was able to solve the problem.It was very difficult for me.I use the tool valgrind to check for memory leaks.At the beginning ,it was showing error.Then,I deleted some unwanted function and it was reduced.

**◎ Attachments**

dse-assignment02.cpp

codeAssignment\_2.txt

**◎ Acknowledgement**

I complete this assignment by myself by using online videos and different books discussing about Algorithm and Data structure.During doing this project,I made a mistake then I solve the consulting with my classmates and teacher.They were all very co-operative.This is a learning curve for me,how to solve this kind of problem in real world .By getting idea from different source and accomplishing the tasks. It was very useful and helpful for me to increase knowledge for solving complex problem.

**◎ Remarks and Grade (by the instructor)**

Instructor Signature:

Grading Date: