header.h

```c
//typedef of structure
typedef struct Polynomial{
    int maxDegree; //the maximum degree of the polynomial
    int *A; //a pointer to a series of integers representing the coefficients
} Polynomial;

//function prototypes
void init(Polynomial *p, int maxDegree);
void append(Polynomial *p, int coefficient, int degree);
void display(Polynomial p);
Polynomial add(Polynomial p1, Polynomial p2);
Polynomial substract(Polynomial p1, Polynomial p2);
Polynomial multiply(Polynomial p1, Polynomial p2);
```

logic.c

```c
#include <stdio.h>
#include <stdlib.h>
#include "./header.h"

//macro for MAX of two elements irrespective of datatype
#define MAX(a, b) ((a) > (b) ? (a) : (b))

//function to initialize the polynomial
void init(Polynomial *p, int maxDegree){
    p->maxDegree = maxDegree; //set the maxDegree
    p->A = (int *) calloc(maxDegree + 1, sizeof(int)); //use calloc to initialize with zeroes
}
```

```c
//function to append a term
void append(Polynomial *p, int coefficient, int degree){
    //check for invalid condition or if coefficient is zero
    if (degree > p->maxDegree || degree < 0 || coefficient == 0) return;
    //add the coefficient to the corresponding degree's coefficient
    p->A[degree] += coefficient;
    return;
}
```

```c
//function to display the polynomial
void display(Polynomial p){
    int i; //for iteration
    int maxDegree = p.maxDegree; //get the maxDegree

    for(i = maxDegree; i >= 0; i--){ //traverse through entire array
        if(p.A[i] == 0) continue; // if coefficient is 0 we dont print

        if(i == 0){ // if degree is zero we dont print the variable
            printf("(%d)\n",p.A[i]);
            return;
        }else if(p.A[i] == 1){ // if coeffcient is 1
            if (i == 1){ //if degree is also one we just print variable x
                printf("x + ");
            }else{
                printf("x^(%d) + ", i); //else just print x with the corresponding degree
            }

        }else if (i == 1){
            printf("(%d)x + ", p.A[i]); //if degree is one we dont show power using '^'
        }else{
            printf("(%d)x^(%d) + ", p.A[i], i); //general case (coeff)x^(degree)
        }
    }

    printf("\b\b \n"); //remove ending '+'
}
```

```c
//function to get entire polynomial at once from user
void getPolynomial(Polynomial *p, int maxDegree){
    int coefficient;
    for(int i = maxDegree; i >= 0; i--){ //ask coefficient for each degree
        printf("Enter Coefficient of x^(%d)\n", i);
        scanf("%d", &coefficient);
        append(p, coefficient, i); //append the term
    }

    return;
}
```

```c
//fill polynomial with random coefficients
void randomPolynomial(Polynomial *p, int maxDegree){
    int coefficient;
    for(int i = maxDegree; i >= 0; i--){
        coefficient = rand() % 100 + 1; //get random coefficient from 1 to 100
        append(p, coefficient, i); //append the term
    }

    return;
}
```

```c
//function to add two polynomials
Polynomial add(Polynomial a, Polynomial b){
    int maxA = a.maxDegree, maxB = b.maxDegree; //get maxDegrees of both polynomials
    int maxDegree = MAX(maxA, maxB); //the maxDegree of result will be the max of degrees of the two
polynomials
    Polynomial result;
    init(&result, maxDegree); //initialize the result polynomial
    int i = maxA, j = maxB;
    while (i >= 0 && j >= 0){
        if (i == j){ //if terms with same degree are present for both polynomials append term to
result with addition of coefficients of the two terms
            append(&result, a.A[i] + b.A[j], i);
            i--;
            j--;
        }else if (i > j){
            append(&result, a.A[i], i); //add term from polynomial where the term with degree exists
            i--;
        }else{
            append(&result, b.A[j], j); //add term from polynomial where the term with degree exists
            j--;
        }
    }

    //add remaining terms (if any left)
    while (i >= 0){
        append(&result, a.A[i], i);
        i--;
    }

    //add remaining terms (if any left)
    while (j >= 0){
        append(&result, b.A[j], j);
        j--;
    }

    return result;
}
```

```c
//function to substract two polynomials
Polynomial substract(Polynomial a, Polynomial b){
    int maxA = a.maxDegree, maxB = b.maxDegree; //get maxDegrees of both polynomials
    int maxDegree = MAX(maxA, maxB); //the maxDegree of result will be the max of degrees of the two polynomials
    Polynomial result;
    init(&result, maxDegree);  //initialize the result polynomial
    int i = maxA, j = maxB;
    while (i >= 0 && j >= 0){
        if (i == j){ //if terms with same degree are present for both polynomials append term to result with substraction of coefficients of the two terms
            append(&result, a.A[i] - b.A[j], i);
            i--;
            j--;
        }else if (i > j){
            append(&result, a.A[i], i); //add term from first polynomial where the term with degree exists
            i--;
        }else{
            append(&result, -b.A[j], j);//add negation of term from second polynomial where the term with degree exists
            j--;
        }
    }

    //add remaining terms from first polynomial (if any left)
    while (i >= 0){
        append(&result, a.A[i], i);
        i--;
    }

    ///add negation of remaining terms from second polynomial (if any left)
    while (j >= 0){
        append(&result, -b.A[j], j);
        j--;
    }

    return result;
}
```

```c
//function to multiply two polynomials
Polynomial multiply(Polynomial a, Polynomial b) {
    int maxA = a.maxDegree, maxB = b.maxDegree; //get maxDegrees of both polynomials
    int maxDegree = maxA + maxB; //the maxDegree of result will be the sum of degrees of the two polynomials
    Polynomial result;
    init(&result, maxDegree);  //initialize the result polynomial

    // multiply each term of first polynomial with each term of the second polynomial and append it to the result
    for (int i = 0; i <= maxA; i++) {
        for (int j = 0; j <= maxB; j++) {
            int newCoeff = a.A[i] * b.A[j];
            int newDegree = i + j;
            append(&result, newCoeff, newDegree);
        }
    }

    return result;
}
```

main.c

```c
#include "./logic.c"

int main() {
    Polynomial p1, p2, sum, difference, product;
    init(&p1, 4);
    init(&p2, 5);
    getPolynomial(&p1, 4);
    randomPolynomial(&p2, 5);
    printf("Polynomial 1: ");
    display(p1);
    printf("Polynomial 2: ");
    display(p2);

    sum = add(p1, p2);
    printf("Sum: ");
    display(sum);

    difference = substract(p1, p2);
    printf("Difference: ");
    display(difference);

    product = multiply(p1, p2);
    printf("Product: ");
    display(product);
    return 0;
}
```

Output

```
Polynomial Using Array ADT>gcc .\main.c -Wall -o main
Polynomial Using Array ADT>.\main.exe
Enter Coefficient of x^(4)
12
Enter Coefficient of x^(3)
23
Enter Coefficient of x^(2)
31
Enter Coefficient of x^(1)
43
Enter Coefficient of x^(0)
15
Polynomial 1: (12)x^(4) + (23)x^(3) + (31)x^(2) + (43)x + (15)
Polynomial 2: (42)x^(5) + (68)x^(4) + (35)x^(3) + x^(2) + (70)x + (25)
Sum: (42)x^(5) + (80)x^(4) + (58)x^(3) + (32)x^(2) + (113)x + (40)
Difference: (-42)x^(5) + (-56)x^(4) + (-12)x^(3) + (30)x^(2) + (-27)x + (-10)
Product: (504)x^(9) + (1782)x^(8) + (3286)x^(7) + (4731)x^(6) + (5502)x^(5) + (4466)x^(4) + (3313)x^(3) + (3800)
x^(2) + (2125)x + (375)
Polynomial Using Array ADT>
```