

CS19003: Programming & Data Structure Lab, Section 15
Autumn 2020-21
Lab Test 3, March 4, 2021

PART - 1

Time: 9-10 am to 10-30 am

Instructions (Read carefully)

1. Your C programs must first contain comments with your name, roll no., and Labtest no. (=3), as done in class.
 2. Name your C file LT3_1_<your roll no>.c for (For example LT3_1_20ME30006.c)
 3. Submit through the links (Intermediate and Final) for PART 1 in moodle. MAKE SURE TO VERIFY YOUR SUBMISSION after final submission.
 4. **(Very Important)** There are two sets of problems. Problem set 1 is for 100% marks and covers the entire syllabus of what is covered in the lab. Problem set 2 is for 60% marks and covers only up to arrays and functions. You are free to choose which set you will attempt. However, there cannot be any mixing between the sets, you can answer only one of the sets. **Submit only one C file for one set.**
 5. All other instructions regarding lab test sent earlier in the slides to be followed strictly.
-

PROBLEM SET 1 (FOR 100% (= 10) MARKS)

Define a structure **TERM** to represent one term of a polynomial. The structure will have two fields, **degree** (integer) and **coeff** (integer) to represent the degree of the term and the coefficient of the term respectively.

Define a structure **POLY** to represent a polynomial. The structure will have two fields, **num** (integer) to represent the number of non-zero coefficient terms in the polynomial, and a pointer **t_list** to an array of **TERM** structures. The array will contain **only the terms with non-zero-coefficients** (so the number of **TERM** structures in the array is given by the field **num**). Also, the terms stored in the array need not be in any particular order of degree (so it is possible that the first element is for the term with degree 3, the next one for the term with degree 1, next one for degree 5, and so on). **There is no upper limit on the number of terms that can be there in a polynomial.**

For example, the polynomial $x^5 + 2x^3 + 7x^2 + 4$ can be represented in a **POLY** structure **P** with **P.num** = 4 (the number of terms with non-zero coefficients), and **P.t_list** pointing to the array (3, 2), (0, 4), (2, 7), (5, 1) where (3,2) denotes the term with degree 3 and coefficient 2 and so on. **Note that the elements of the array are not in sorted order of degree (they may be or may not be, so you cannot assume any sorted order in your program).**

Write a function

struct POLY *AddPoly(struct POLY P1, struct POLY P2)

that takes two polynomials **P1** and **P2**, and returns a pointer to a **POLY** structure containing the addition of the two polynomials. **In the **POLY** structure returned, the terms in the array pointed to by**

the pointer **t_list** must be in decreasing order of degree (so first element will be the term with the highest degree, the next element will be the term with the next highest degree, and so on). Also, the array in the **POLY** structure returned must be allocated with size exactly equal to the number of terms to be stored there (number of non-zero coefficient terms in the sum of the two polynomials). **P1** and **P2** must not be changed in any way. You cannot use any additional arrays (3 marks penalty if you do).


Write a function

struct POLY *MultPoly(struct POLY P1, struct POLY P2)

that takes two polynomials **P1** and **P2**, and returns a pointer to a **POLY** structure containing the product of the two polynomials. In the **POLY** structure returned, the terms in the array pointed to by the pointer **t_list** must be in decreasing order of degree (so first element will be the term with the highest degree, the next element will be the term with the next highest degree, and so on). Also, the array in the **POLY** structure returned must be allocated with size exactly equal to the number of terms to be stored there (number of non-zero coefficient terms in the product of the two polynomials). **P1** and **P2** must not be changed in any way. You cannot use any static arrays, but you can use additional dynamically allocated arrays if needed for this function.

Do not write any other functions other than **AddPoly()**, **MultPoly()**, and **main()**.

Write a **main()** function that does the following in exactly this order:

1. Define two variables **P** and **Q** of type **struct POLY**.
2. Read the number of terms of non-zero coefficients, **n**, of the first polynomial **P**.
3. Malloc the array in **P** to store the **n** terms.
4. Read the **n** terms of **P**. For each term, read the degree first and then the co-efficient. The input need not be given in sorted order of degree.
5. Read the number of terms of non-zero coefficients, **m**, of the second polynomial **Q**.
6. Malloc the array in **Q** to store the **m** terms.
7. Read the **m** terms of **Q**. For each term, read the degree first and then the co-efficient. The input need not be given in sorted order of degree.
8. Print the two polynomials nicely (assume that the polynomial is in variable **x**. See the sample output given .
9. Call the function **AddPoly()** to add **P** and **Q**. Print the polynomial returned.
10. Call the function **MultPoly()** to multiply **P** and **Q**. Print the polynomial returned.

Example:

$$P = x^5 + 2x^3 + 7x^2 + 4, \quad Q = 2x^7 + 10x^5 + 11x^3 + 9x + 4$$

Your program should output (exact messages can vary):

$$P = x^5 + 2x^3 + 7x^2 + 4$$

$$Q = 2x^7 + 10x^5 + 11x^3 + 9x + 4$$

$$\text{After addition, new polynomial is: } 2x^7 + 11x^5 + 13x^3 + 7x^2 + 9x + 8$$

$$\text{After multiplication, the polynomial is: } 2x^{12} + 14x^{10} + 14x^9 + 31x^8 + 78x^7 + 31x^6 + 121x^5 + 18x^4 + 115x^3 + 28x^2 + 36x + 16$$

PROBLEM SET 2 (FOR 60% (= 6) MARKS)

Consider a polynomial in a single variable x represented by an integer n that stores the highest degree of the polynomial, and an integer array P that stores the integer coefficients of all terms of the polynomial, in decreasing order of degree (coefficient of highest degree term at index 0, next higher degree at index 1, and so on). If a certain term is not present in the polynomial, its coefficient will be stored as 0. Assume that all coefficients are 0 or positive integers.

Write a C function

int Find3rdMaxCoeff(int P1[], int n1, int P2[], int n2)

that takes as parameters $P1$ and $n1$ representing the coefficients and the highest degree of one polynomial (in the above form), and $P2$ and $n2$ representing the coefficients and the highest degree of a second polynomial. The function returns the coefficient of the third highest degree term (third highest among terms with non-zero coefficients only) in the product of the two polynomials. If there is no third highest degree term (only two degrees are present with non-zero coefficients), the function should return -1. **You cannot use any additional array** (1 mark penalty if you do).

Write a C function

int FindMissing(int P1[], int n1, int P2[], int n2)

that takes as parameters $P1$ and $n1$ representing the coefficients and the highest degree of one polynomial (in the above form), and $P2$ and $n2$ representing the coefficients and the highest degree of a second polynomial. The function returns the minimum degree of a missing term (a term with zero coefficient) in the product of the two polynomials. Note that the product may not have any missing term. In that case, the function should return -1. **You cannot use any additional array** (2 marks penalty if you do).

Write a **main()** function that does the following:

1. Reads in the highest degree of the first polynomial in a variable $n1$ (< 100).
2. Reads in the coefficients of the first polynomial in an array P , in descending order of degree (you should enter 0 as coefficient of missing terms).
3. Reads in the highest degree of the second polynomial in a variable $n2$ (< 100).
4. Reads in the coefficients of the second polynomial in an array Q , in descending order of degree (you should enter 0 as coefficient of missing terms).
5. Prints the two polynomials nicely (assume that the polynomial is in variable x . See the sample output given).
6. Calls **Find3rdMaxCoeff()** to find and print the coefficient of the third highest degree term in the product of the two polynomials. Print an appropriate message if there is no third highest degree term (function returned -1).
7. Calls **FindMissing()** to find and print the minimum degree of a missing term in the product of the two polynomials. Print an appropriate message if there is no missing term (function returned -1).

Example:

$$\mathbf{P} = x^5 + 2x^3 + 7x^2 + 4, \mathbf{Q} = 2x^7 + 10x^5 + 11x^3 + 9x + 4$$

Your program should output (exact messages can vary):

$$P = x^5 + 2x^3 + 7x^2 + 4$$

$$Q = 2x^7 + 10x^5 + 11x^3 + 9x + 4$$

The coefficient of the third highest degree term in the product is 14

The minimum degree of a missing term in the product is 11