Programming Lab 1 CS506

Read the assignment carefully. There are 2 exercises in this assignment; 2 programs to be submitted.

- Single .c file for each exercise. Program must be compilable using gcc as we will be likely using linux platform to evaluate your assignments
- You MUST follow the Filename format :
  - P01a_<Your First name>_<Entrynumber>_CS506_2021.c for part A,
    - For e.g. P01a_Raman_2021CSB9999_CS506_2021.c for part A exercise
  - P01b_<Your First name>_<Entrynumber>_CS506_2021.c for part B,
    - For e.g.P01b_Raman_2019CSB9999_CS506_2021.c for part B exercise
- Input outputs that you tried or used for testing your code can be included in your program file at end within comments (using /* */).
- If you know you code is not working for some cases etc, do mention that clearly in comments section at the beginning of your code (using /* ….. */)
- Deadline are as follows:
  - For parts A and B: 4th Sep 2021, 11:59:59 PM
  - While submitting, exercise due care and caution to submit the appropriate code (your last updated program, not your intermediate trial codes) and with proper filename.
  - Instructions how and where to submit the assignments on Moodle/via Google Form or Google Classroom will be shared with you soon.
- No Plagiarism Please. Please note that we may do plag check now or later even after endsem, and you will be then (severely) penalized. You must do your own coding and neither take nor provide codes from/to anyone else.
  - Those who complete their works, they are kindly requested to not share their codes in name of friendship or so. You may get severely penalized for the same and by sharing codes, you are not really helping them either.
  - We sincerely request and hope that you will try your level best and not adopt any unfair means so that we need NOT go through the painful experience of penalizing you.

//Note: We will also share with you some sample input output files and you are requested to follow the proper input output format, As we will be using similar files and the commands of the following nature (in some automated or semi-automated way) to evaluate your submitted program:

```
$gcc P01a_Raman_2019CSB9999.c -o p01a_2019CSB9999

$./p01a_2019CSB9999 < p01a_input1 > p01a_2019CSB9999_output1

$./p01a_2019CSB9999 < p01a_input2 > p01a_2019CSB9999_output2

$./p01a_2019CSB9999 < p01a_input3 > p01a_2019CSB9999_output3

$diff p01a_2019CSB9999_output1 p01a_output1

$diff p01a_2019CSB9999_output2 p01a_output2

$diff p01a_2019CSB9999_output3 p01a_output3
```

========================EXERCISE A (Using Linked List concepts) ========================

Using linked list, write a program to multiply and evaluate polynomials: P1(x), P2(x), and P3(x) where P1(x) and P2(x) are input polynomials and P3(x) = P1(x)*P2(x). Each node in the linked list correspond to a term in the polynomial. So, in your node structure - you may have two data components – int pow, coeff ; and one pointer to the next node.

Input Format:

First line mentions K i.e. the number of test cases. Then there are three lines for each test case, In the first two lines of a test case, First number indicate the highest degree of polynomials N and then there are N+1 integers which are the coefficients of polynomial terms in descending order.

In the third (and last line) of a test case, there is one integer i.e. value of x for which you need to evaluate the polynomials. (Constraints: $0 \le K \le 50$, $0 \le N \le 9$, $-2 \le x \le 2$, and Input coefficient terms would be between -100 to +100; Assume you can safely do calculations for each polynomial term without worrying about underflow/overflow issues).

Sample Input 1:

```
1
7 1 0 0 0 10 -3 0 1
3 4 0 0 -2
2
```

Explanation of Input Format (Considering 2nd Polynomial mentioned above)

3	4	0	0	-2
Highest degree of polynomial	Coefficient of x^3	Coeff of x^2	Coeff of x^1	Coeff of x^0

As per above format: 7 1 0 0 0 10 -3 0 1===> $x^7 + 10x^3 - 3x^2 + 1$
3 4 0 0 2 ===> $4x^3 - 2$
2 ===> value of x should be in range of -2 to 2.

Sample Output 1:

```
P1(x) : 1x^7 + 10x^3 - 3x^2 + 1
P2(x) : 4x^3 - 2
P3(x) : 4x^10 – 2x^7 + 40x^6 - 12x^5 - 16x^3 + 6x^2 - 2
P1(2) = 197
P2(2) = 30
P3(2) = 5910
```

Note: There is a single space before and after =, +, -, and : in the output.

Explanation of Output Format:

4x^10 – 2x^7 + 40x^6 - 12x^5 - 16x^3 + 6x^2 - 2 ====> $f_1(x)*f_2(x)$
197 30 5910 ====> $f_1(2)$ $f_2(2)$ $(f_1(2) * f_1(2))$

Note that there should be **NO** nodes in the linked list for the polynomial terms having 0 coefficient value. In the above example, the polynomials f1 and f2 were to be represented using 4 and 2 nodes in the linked list (created after reading inputs), respectively.

Sample Input 2	**Sample Output 2**

2
7 1 0 0 0 10 -3 0 1
3 4 0 0 -2
2
7 1 0 0 0 10 -3 0 1
3 4 0 0 -2
1
	P1(x) : 1x^7 + 10x^3 - 3x^2 + 1
	P2(x) : 4x^3 - 2
	P3(x) : 4x^10 – 2x^7 + 40x^6 - 12x^5 - 16x^3 + 6x^2 - 2
	P1(2) = 197
	P2(2) = 30
	P3(2) = 5910
	P1(x) : 1x^7 + 10x^3 - 3x^2 + 1
	P2(x) : 4x^3 - 2
	P3(x) : 4x^10 – 2x^7 + 40x^6 - 12x^5 - 16x^3 + 6x^2 - 2
	P1(1) = 9
	P2(1) = 2
	P3(1) = 18

Remarks: There are two test cases in 2nd Sample Input.

Remember to appropriate free the memory space after each test case.

// Input would be read from terminal i.e. Stdin and not from any file.

=========================EXERCISE B (Use Linked List concepts) =========================

Consider a new function # (somewhat related to factorial !) that is defined for any non-negative integer **n** as follows:

n# = (1^1) * (2^2) * (3^3) * ....* ((n-1) ^ (n-1)) * (n^n)

//Remember that **n!** = 1*2*3*4*…..*(n-1)*n and is different **0# = 1 1#=1 2#=4 2!=2**

Given a non-negative number n and positive number k (atmost three digit), Write a program in C that computes and provides the value of n# ,
Z_n i.e. the number of end zeroes in n# value, and
$Q_{n,k}$ i.e. the number of times k pattern appears in n#.

Constraints and Sample input/output -

First line of the input file mentions positive value T that denotes the number of test cases. Then follows T lines and each of them mentions two values (single space separated) n_i and k_i
Constraints/Assumptions:
All these input values are non-negative.
T<500, n_i <= 200 and k_i <= 999

Input would be read from terminal i.e. Stdin and not from any file.

Input Format: T N_1 k_1 N_2 k_2 ... ... N_T k_T	**Output format (T lines each having 3 values, single space separated):** Z_{N1} $Q_{N1,k1}$ N_1# Z_{N2} $Q_{N2,k2}$ N_2# ... ... Z_{NT} $Q_{NT,kT}$ N_T#
Sample Input: 6 3 1 5 7 9 941 9 09 7 3 9 22	**Sample Output** 0 1 108 5 0 86400000 5 2 2157794122294185620916802682880000 5 1 2157794122294185620916802682880000 5 3 331976639877120000 5 2 2157794122294185620916802682880000

//Note: 22 is appearing 4 times in 222221111000

======