Lab 1 CS201 2019

Read the assignment carefully. There are 3 exercises in this assignment; 3 programs to be submitted.

- Single .c file for each exercise. Filename format : L01a_<First name>_<Entrynumber>_CS201.c for part A, Similarly for other parts. E.g. L01b_Raman_2018CSB9999_CS201.c
- Input outputs that you tried or used for testing your code can be included in your program file at end within comments (using /* */). Also mention your name, entry number and self declaration regarding doing assignment yourself / whatever assistance you took from others.
- Deadline would be announced soon in class and via email. Will also explain about Exercise C.

==========EXERCISE A (Using 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!** is different and **n!** = 1*2*3*4*…..*(n-1)*n **0# = 1 1#=1 2#=4**

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 screen 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: 5 3 1 5 7 9 941 9 09 7 3	**Sample Output** 0 1 108 5 0 86400000 5 2 215779412229418562091680268288000000 5 1 215779412229418562091680268288000000 5 3 3319766398771200000

==========EXERCISE B (Using Stacks/Queue concepts) ==========

Write a C program to compute the expressions (as in C) having operators and integer values operands. Input first line would mention T i.e. number of queries and then follows T rows each for an expression having integers, operators and symbol #. The symbol # is used to denote end of expression and you can neglect anything that appears in that row after symbol #.

Other constraints:

Max 20 operators. Integers range : -1000 to 1000
used to mark the end of query. Ignore any symbol after #.
No brackets etc. No operators other than what mentioned below would be considered valid.
Operands and Operators are single-space separated.
Some operators may consist of two characters e.g. >> (there is no space between these characters)
Note: &, ^, | are bitwise operators

Query Response:
Value of the evaluated expression
If expression was invalid, mention "Error" without quotes

Operators	Associativity
* / %	left to right
+ -	left to right
<< >>	left to right
< <= > >=	left to right
== !=	left to right
&	left to right
^	left to right
|	left to right

Sample Input:	Sample Output
4 1 + 2 * 3 # 1 + 3 – 5 # 4 + 3 & 5 <= 18 # 4 + 3 * 5 <= 18 #	7 -1 1 0

==========EXERCISE C (Using Stacks/Queue concepts) ==========

Given N and k, write program that computes the number of stack permutations possible with numbers 1, 2, …. N and determines the k^{th} sequence in these stack permutations, considering lexicographic ordering. Input first line would mention T i.e. number of queries and then follows T rows, each row mentioning N and k. (Constraints N<= 16)

Query Response:
Total number of stack permutations possible with these N different numbers,
and then the k^{th} stack permutation.

Sample Input:	Sample Output
3 3 2 3 5 2 2	5 1 3 2 5 3 2 1 2 2 1

==========