Data Structures Assignment2

The assignment deadline is Saturday 5pm. The O.J. Will be up soon, till then start solving the problems

Questions:

Question1 : Radix Sort using Linked Lists

Given N numbers, Sort the numbers using a radix sort.

First line contains the number of test cases(T). For each test case, first line contains the number of elements(N) followed by a line of N elements and another line containing a query(K). For each Kth pass of the algorithm, print the elements of radix sort(iterations starting from 0). Output for each query will be on a separate line. Iterate over each bucket from the beginning and output every element.

Limits:

1<N<10^6

1<Ni<10^50 (each of the numbers can have 50 digits)

Sample input:

```
1
10
64 8 216 512 27 729 0 1 343 125
1
```

Sample output:

```
0 1 8 512 216 125 27 729 343 64
```

Question2 : Infix to Prefix Conversion:

Given an Infix Expression, convert it into an Prefix Expression.

First line contains T, the number of test cases. Followed by T lines of infix expression(s). Convert each of them to their corresponding prefix expression and print each of the expression on separate lines.

Other Criterions

i) All the operands are single digit numbers.

ii) Precedence of the operators are as follows:

'+' and '-' : 1
'/' and '*' : 2
'^' : 3
'(' and ')' : 4

iii) For all operators associativity is left to right, except for exponentiation associativity is right to left.

iv) If an expression is not a valid infix expression then print invalid.

v)

0<Length of the expression: <=100

Sample Input:

```
7
(3+5^7)*4+2^6
3^7^5
((3+5)*(9-5))+(6-7)/8
4^6++5-3*3/9
((3*5)+(7/9))
```

((3*(5+7))/9)
(3*(5+(7/9)))

Sample Output:
+*+3^574^26
^3^75
+*+35-95/-678
invalid
+*35/79
/*3+579
*3+5/7

Question3 : Prefix Evaluation

Given a Prefix Expression. Evaluate and output the result. All operators to be considered(+ , - , / , * , ^ , '(' , ')') // '^' → Exponent.

First line contains the number of test cases(T). Followed by T lines of prefix expressions. For each of the prefix expression, output the evaluation of it on separate lines. If the given expression is an invalid prefix expression print it as "invalid" (without quotes).

Sample Test Case:

Sample Input:
1
*23

Sample Output:
6

Question4 : Polynomial ADT :: Addition, Subtraction and Multiplication

Given a Sparse Polynomial. Perform the operations.1-Addition, 2-Subtraction, 3-Multiplication. First line contains the number of test cases. Following by T test cases. Each test case has M and N in the first line(M-> number of terms in first polynomial and N is the number of terms in the second polynomial) Followed by M+N lines, each having A and B, A-> power of the term and B-> coeffecient. First M lines for first and next N lines for second polynomial. Output the addition, subtraction and multiplication for each of the test case.(first+second, first-second, first*second). For each of addition, subtraction and multiplication first line contains the number of terms(R) in the resultant expression. Followed by R lines of A and B (A-> power of the term B-> coeffecient of the term).

Question5:
The lucky one!

A university has N guys, and one of them needs to be chosen. The chooser being fond of numbers, asks the guys to queue in order of increasing heights. Now, arranges them in circular order by bringing together the tallest and the shortest one.

The chooser now chooses a number K, and plays the following game.

Starting with the shortest guy, and moving in the direction of non-decreasing heights, asks the Kth guy to leave the circle. Now starting with the next guy (K + 1 in the initial circle), finds the Kth guy and asks him to leave. This process is continued, until only one remains. And he’s the lucky one.

This process might be fun to do on the university ground, but will take a lot of time and space. Being a computer scientist, write a code that does this for you.

Input :
N
H0 H1 H2 ..Hi… HN
K

where
N : Number of guys, 1<=N<=1000
Hi : Height of ith guy. 1<=Hi<=10^9
K : Chooser’s Number 1<=K<=1000

Output :
Hk

where Hk : Height of the guy last remaining.

Sample Case 1 :
Input :
3
5 1 3
2

Output :
5

Explanation : After being arranged in circular order : 1->3->5->(1) (as in the next of 5 is 1). Counting is started with the shorted guy, in the current case, the guy with height 1, and Kth (2nd) guy is asked to leave the circle (guy with height 3). The new circle become : 1->5->(1). Counting now starts with the guy of height 5, as he is next to the removed one. The 2nd guy with height 1 is asked to leave the circle. The circle now has only 1 element. 5->(5).