PREVIOUS TCSCODEVITA- 5 PROBLEMS

PROBLEM 1- kth largest factor of N Problem

Problem1

A positive integer d is said to be a factor of another positive integer N if when N is divided by d, the remainder obtained is zero. For example, for the number 12, there are 6 factors 1, 2, 3, 4, 6, 12. Every positive integer k has at least two factors, 1 and the number k itself.

Given two positive integers N and k, write a program to print the kth largest factor of N.

Input
The input is a comma separated list of positive integer pairs (N, k)
Output
The kth highest factor of N. If N does not have k factors, the output should be 1.
Constraints
1<N<10000000000. 1<k<600
You can assume that N will have no prime factors which are larger than 13.

Example 1
Input:
12,3
Output:
4
Explanation:
N is 12, k is 3. The factors of 12 are (1,2,3,4,6,12). The highest factor is 12 and the third largest factor is 4. The output must be 4

Example 2
Input:
30,9
Output:
1
Explanation: N is 30, k is 9. The factors of 30 are (1,2,3,5,6,10,15,30). There are only 8 factors. As k is more than the number of factors, the output is 1.

Test Cases

SNo	Input	Output
1	16,2	8
2	25,1	25
3	12,3	4
4	30,9	1
5	15,3	3

6	12,12	1
7	18,3	6

PROBLEM 2 —-Minimum Product Array Problem

Problem2 :

The task is to find the minimum sum of Products of two arrays of the same size, given that k modifications are allowed on the first array. In each modification, one array element of the first array can either be increased or decreased by 2.

Note- the product sum is Summation (A[i]*B[i]) for all i from 1 to n where n is the size of both arrays

Input Format:

First line of the input contains 2 integers n and k delimited by whitespace

Second line contains the Array A (modifiable array) with its values delimited by spaces

Third line contains the Array B (non-modifiable array) with its values delimited by spaces

Output Format:

Output the minimum sum of products of the two arrays

Sample Input 1:

3 5

1 2 -3

-2 3 -5

Sample Output 1:

-31

Explanation for sample Input and Output 1:

Here total numbers are 3 and total modifications allowed are 5. So we modified A[2], which is -3 and increased it by 10 (as 5 modifications are allowed). Now final sum will be

(1 * -2) + (2 * 3) + (7 * -5)

-2 + 6 - 35

-31

-31 is our final answer.

Sample Input 2:

5 3

2 3 4 5 4

3 4 2 3 2

Sample Output 2:

25

SNo	Input	Output
1	4 3 2 5 6 1 7 5 3 1	16
2	3 2 5 3 6 -2 5 -6	-55
3	3 5 1 2 -3 -2 3 -5	-31
4	5 3 2 3 4 5 4 3 4 2 3 2	25

SNo	Input	Output
5	2 1 1 2 2 1	0

PROBLEM 3 – Bank Compare Problem

Problem3:

There are two banks; Bank A and Bank B. Their interest rates vary. You have received offers from both bank in terms of annual rate of interest, tenure and variations of rate of interest over the entire tenure.

You have to choose the offer which costs you least interest and reject the other.

Do the computation and make a wise choice.

The loan repayment happens at a monthly frequency and Equated Monthly Installment (EMI) is calculated using the formula given below :

EMI = loanAmount * monthlyInterestRate /

(1 - 1 / (1 + monthlyInterestRate)^(numberOfYears * 12))

Constraints

1 <= P <= 1000000

1 <=T <= 50

1<= N1 <= 30

1<= N2 <= 30

Input Format

First line : P – principal (Loan Amount)

Second line : T – Total Tenure (in years).

Third Line : N1 is number of slabs of interest rates for a given period by Bank A. First slab starts from first year and second slab starts from end of first slab and so on.

Next N1 line will contain the interest rate and their period.

After N1 lines we will receive N2 viz. the number of slabs offered by second bank.

Next N2 lines are number of slabs of interest rates for a given period by Bank B. First slab starts from first year and second slab starts from end of first slab and so on.

The period and rate will be delimited by single white space.

Output

Your decision – either Bank A or Bank B.

Explanation
Example 1
Input
10000
20
3
5 9.5
10 9.6
5 8.5

3
10 6.9
5 8.5
5 7.9

Output
Bank B
Example 2
Input
500000
26
3

13 9.5
3 6.9
10 5.6

3
14 8.5

6 7.4
6 9.6

Output
Bank B

SNo	Input	Output
1	500000 29 3 9 6.9 5 8.5 15 7.0 3 10 10.5 10 9.6 9 8.5	Bank A
2	500000 26 3 13 9.5 3 6.9 10 5.6 3 14 8.5 6 7.4 6 9.6	Bank A
3	100000 25 3 10 8.5 7 12.4 8 9.6 3 13 10.5 6 7.9 3 9.6	Bank B
4	10000 20 3 5 9.5 10 9.6 5 8.5 3 10 6.9	Bank B

SNo	Input	Output
	5 8.5 5 7.9	
5	100000 28 3 16 10.5 4 7.9 8 9.6 3 14 8.5 7 12.4 12 9.6	Bank A

PROBLEM 4 — Recurring Deposit Problem

Problem4:

Manisha is a smart investor. She has opened a Recurring Deposit (RD) account with a bank. She has negotiated smartly with the bank. Bank has agreed to vary the interest rate on her RD account such that she is always able to comfortably beat inflation. Your task is to calculate maturity value of her RD account in face of varying interest rates that she will enjoy.

Input Format:

First line contains principle amount P
Second line contains original rate of interest per annum R
Third line contains tenure in months T
Next few lines contain a tuple with 3 values. Each tuple contains delimited by whitespace, where updated _rate is applied on From_month and To_month, both inclusive.
-1 shows the end of input

Output Format:

Print the maturity amount after specified tenure or month in the format "Final Amount : "

Constraints:

P > 0 ; it can be float value
R >=0 ; it can be float value
T >0 ; it can be integer only
Calculation should be done upto 11-digit precision
Maturity amount should be printed to its nearest integer value

Sample Input:

5000
10
12
3 7 11
8 12 12
-1

Sample Output:

63771

Explanation:

For every month, Manisha will deposit 5000. So, principal = principal + 5000.

interest = interest + principal * rate / 100 / T

Intially,
principal = 0
interest = 0

For 1st month,
principal = 0 + 5000 = 5000
There are no changes in rate for 1st month. So, rate = 10
interest = 5000 * 10 / 100 / 12 = 41.66666666667

For 2nd month,
principal = 5000 + 5000 = 10000
There are no changes in rate for 2nd month. So, rate = 10
interest = 41.66666666667 + 10000 * 10 / 100 / 12 = 125

For 3rd month,

principal = 10000 + 5000 = 15000
Rate should be changed to 11 for 3rd month to 7th month. So, rate = 11
interest = 125 + 15000 * 11 / 100 / 12 = 262.5

Similarly after 12 months, principal will be 60000 and interest will be 3770.83333333333. Their sum will be the final amount after rounding(63771).

SNo	**Input**	**Output**
1	1000 9 12 5 12 10 -1	12642
2	2565.50 7.5 12 6 12 8 -1	32104
3	5000 10 12 3 7 11 8 12 12 -1	63771
4	200.75 6.9 5 -1	1021
5	7500 7.5 12 4 6 8 7 9 9 9 12 10 -1	94238
6	3250.75 10 5 -1	16660

PROBLEM 5 - Compiler Design - Limit the Method Instructions Problem

Raj is a newbie to the programming and while learning the programming language he came to know the following rules:
- Each program must start with '{' and end with '}'.
- Each program must contain only one main function. Main function must start with '<' and end with '>'.
- A program may or may not contain user defined function(s). There is no limitation on the number of user defined functions in the program. User defined function must start with '(' and end with ')'.
- Loops are allowed only inside the functions (this function can be either main function or user defined function(s)). Every loop must start with '{' and end with '}'.
- User defined function(s) are not allowed to be defined inside main function or other user defined function(s).
- Nested loops are allowed.
-Instructions can be anywhere inside the program.
-Number of instructions inside any user defined function must not be more than 100.

If any of the above conditions is not satisfied, then the program will generate compilation errors. Today Raj has written a few programs, but he is not sure about the correctness of the programs. Your task is to help him to find out whether his program will compile without any errors or not.

Input Format: Input contains a single line L, where L is a program written by Raj.
Output Format: Print "No Compilation Errors" if there are no compilation errors, else print "Compilation Errors".

Constraints:
L is a text and can be composed of any of the characters {, }, (,), <, >and P, where P will represents the instruction.
L, comprised of characters mentioned above should be single spaced delimited.
Number of characters in the text, |L| < = 10000

Sample Input 1:
{ < > (P) }

Sample Output 1:
No Compilation Errors

Sample Input 2:
{ < { } > ({ }))

Sample Output 2:
Compilation Errors

Sample Input 3:
{ ({ }) }

Sample Output 3:
Compilation Errors

S.No	Input	Output
1	{ P < P { P { P } P } P > P } ( )	Compilation Errors
2	{ < { } > ({ }))	Compilation Errors
3	{ < > P }	No Compilation Errors
4	{ < > ( ) }	No Compilation Errors
5	{ < > ( ) ({ { } }) ({ }) { } }	Compilation Errors
6	{ < > (< >) (P) ( ) }	Compilation Errors
7	{ < { { } } > }	No Compilation Errors
8	{ < P > { P } }	Compilation Errors
9	{ < > ({ }) }	No Compilation Errors
10	{ < P > }	No Compilation Errors
11	{ < { { } } > { } }	Compilation Errors
12	{ < > (P (P) P) }	Compilation Errors
13	{ < > ({ P }) < > }	Compilation Errors

14	{ P < P { P { P } P } P > P }	No Compilation Errors
15	{ < > ( ) ({ } P) (P { }) }	No Compilation Errors
16	{ < ( ) > ( ) ( ) }	Compilation Errors
17	{ < > ( ) ( ) }	No Compilation Errors
18	{ < > ( ) ({ ( ) }) }	Compilation Errors
19	{ < { { P } } > }	No Compilation Errors
20	{ < { P P P } > (< >) }	Compilation Errors
21	{ < {) } > }	Compilation Errors
22	{ < P > P }	No Compilation Errors
23	{ < > ( ) ({ } P) (P { }) } ( )	Compilation Errors
24	{ < > (( )) ( ) }	Compilation Errors
25	{ < > ( ) ({ { } }) ({ }) }	No Compilation Errors
26	{ < > (P) }	Compilation Errors
27	{ < > ( ) (P) ( ) }	No Compilation Errors
28	{ < > ({ { } }) }	No Compilation Errors
29	{ < > (P) }	No Compilation Errors
30	{ < > (P) ({ }) }	No Compilation Errors
31	{ < > ( ) ({ }) ({ }) }	No Compilation Errors
32	{ < > ( ) ({ }) }	No Compilation Errors
33	P { < > }	Compilation Errors

34	{ < > ( ) ( ) ( ) }	No Compilation Errors
35	{ < > }	No Compilation Errors
36	{ < { P P P } > }	No Compilation Errors
37	{ < > (P P P) }	No Compilation Errors
38	{ < P > { } }	Compilation Errors
39	{ < P ( ) P > (P) ( ) }	Compilation Errors
40	{ < ( ) > ({ { } }) }	Compilation Errors
41	{ ({ }) }	Compilation Errors
42	{ < { } > }	No Compilation Errors
43	{ < > (P) ({ }) } < >	Compilation Errors
44	{ < { { P } } > } { }	Compilation Errors
45	{ < > ( ) ({ } }) ({ }) }	Compilation Errors
46	{ < > (P) }	No Compilation Errors
47	{ < > (({ })) }	Compilation Errors
48	{ < P P > (P) ( ) }	No Compilation Errors
49	{ < ( ) > P }	Compilation Errors
50	{ < > ({ P }) }	No Compilation Errors
51	{ ( ) }	Compilation Errors

Problem 6 : Sheldon Cooper and his beverage paradigm Problem

Sheldon Cooper, Leonard Hofstadter and Penny decide to go for drinks at Cheese cake factory. Sheldon proposes to make a game out of this. Sheldon proposes as follows,

To decide the amount of beverage they plan to consume, say X.
Then order for a random number of different drinks, say {A, B, C, D, E, F} of quantities {a, b, c, d, e, f} respectively.
If quantity of any three drinks add up to X then we'll have it else we'll return the order.
E.g. If a + d + f = X then True else False
Your task is to find out if there can be any combination of three beverage sizes that can sum up to the quantity they intend to consume. If such a combination is possible
print True else False

Input Format:

First line contains number of bottles ordered denoted by N
Next N lines, contains a positive integer Ai, the size of the ith bottle
Last line contains the quantity they intend to consume denoted by X in text above
Output Format:
True, if combination is possible
False, if combination is not possible
Constraints:
N >= 3
Ai > 0
1 <= i <= N
X > 0

Sample Input 1:
6
1
4
45
6
10

8
22

Sample Output 1:

True

Sample Input 2:

4
1
3
12
4
14

Sample Output 2:

False

SNo	Input	Output
1	4 1 3 12 4 14	False
2	8 10 20 30 40 50 60 70 80 175	False
3	1 15 15	False

SNo	Input	Output
4	6 1 4 45 6 10 8 22	True
5	3 10 10 10 30	True
6	10 5 5 5 1 2 3 4 6 8 10 15	True