JAVA Winning Camp 2020

Logic Building Exercise Day wise

Day 1: Longest Increasing sequence (LIS)

Given an integer array, find the length of its longest increasing sequence.

The prototype of the function is as below:
public static void longestIncreasingSeq(int input1, int[] input2);
The function takes as input an integer **input1** and an integer array **input2**, containing **input1** number of integers, and sets the **output1** variable to the length of its LIS (longest increasing sequence).

Example 1:
input1 = 9
input2[] = {11,3,4,7,8,12,2,3,7}
output1 should be 5.
Explanation:
In the given array **input2**, the two increasing sequences are "3,4,7,8,12" and "2,3,7". The first sequence i.e. "3,4,7,8,12" is the longer one containing five items. So, the longest increasing sequence = 5.

Example 2:
input1 = 4
input2[] = {1,3,2,1}
output1 should be 2

Explanation:
In the given array **input2**, the increasing sequence is "1,3" containing two items. So, the longest increasing sequence = 2.

Example 3:
input1 = 12
input2[] = {12,17,21,3,7,9,10,11,33,100,4,8}
output1 should be 7

Explanation:
In the given array **input2**, the increasing sequences are "12,17,21" containing three items, "3,7,9,10,11,33,100" containing seven items and "4,8" containing two items. So, the longest increasing sequence = 7
.**Example 4:**
input1 = 6
input2[] = {12,11,10,9,8,7}
output1 should be 0
Explanation:
In the given array **input2**, there is NO increasing sequence. All the items are in decreasing order, hence the longest increasing sequence = 0

Day 2: DigitSum

The labels on a trader's boxes display a large number (integer). The trader wants to label the boxes with a single digit ranging from 1 to 9. He decides to perform digit sum on this large number, continuously till he gets a single digit number.

NOTE: In mathematics, the digit sum of a given integer is the sum of all its digits, (e.g.: the digit sum of 84001 is calculated as 8+4+0+0+1 = 13, the digit sum of 13 is 1+3 = 4).

Write a function (method) that takes as input a large number and returns a single digit by performing continuous digitSum on this number, and on the resulting numbers, till the resulting number is a single digit number in the range 1 to 9.

Example 1: If the large number whose single-digit digitSum is to be found is 976592, the process is as below –

9+7+6+5+9+2 = 38

3+8 = 11

1+1 = 2

Thus, the single-digit digitSum for the number 976592 is 2.

Example 2: If the large number whose single-digit digitSum is to be found is 123456, the process is as below –

1+2+3+4+5+6 = 21

2+1 = 3

Thus, the single-digit digitSum for the number 123456 is 3.

For negative numbers, the result should also be in negative.

Example 3: If the large number whose single-digit digitSum is to be found is -123456, the answer would be -3.

The prototype of the function is:

int digitSum(int input1);

where, **input1** is the large number whose single-digit digitSum is to be found.

The single digit result is expected to be returned by the function

Day 3: Tywins Tactics

Tywin Lannister is a tactical genius. At the heart of his tactical skills, is the way he has organized his armies, and the way he is able to estimate his soldiers' skill-levels, thus helping him make crucial decisions as to whom to dispatch to which areas of the war.

His army is organized in the form of a hierarchy - indeed it is a tree, with him as the root. We say "**A has immediate superior B**" if **A** reports directly to **B**. We further say "**A has superior B**" if there is a chain of soldiers starting with **A**, ending with **B**, and where each soldier reports directly to the next soldier of the chain. Further, each soldier is assigned an initial **skill-level** based on prior experience and battle proficiency.

In order for Tywin to decide whom to send to which battle, he has the following scheme: He chooses a particular soldier **S** as the leader of his temporary 'regiment', and sends in to battle, **S** as well as all the soldiers that have **S** as one of their superiors. He estimates the skill level of the regiment as the total skill level of all the soldiers under **S** (denoted by query "**Q S**").

After a battle, he may want to update the skill levels of some soldiers. If he wants to update the skill level of soldier **S** to value **x**, it is denoted by the query "**U S x**".

You are given the structure of the army, whose size is **N**, the initial skill levels of all the individual soldiers, as well the number of queries **M**. For each query of the type "**Q S**", report the sum of skill-levels of all the soldiers who have **S** as their superior.

Note: The soldiers are numbered 1 to **N**, and Tywin is given the number **1**.

Input

The first line consists of the integers **N** and **M**, denoting the number of soldiers and the number of queries.
This is followed by a single line consisting of **N** nonnegative values - the **skill levels of the N soldiers**.
This is then followed by **N-1** lines consisting of pairs of integers **(u, v)**, denoting that either **u is an immediate superior of v, or vice-versa**.
Finally you are given the **M** queries, of either the form "**U S x**" or "**Q S**".

Output

For each "**Q S**" query, output the sum of skill values of all the soldiers under **S**.

Constraints

- $\mathbf{1 \leq N \leq 10^5}$
- $\mathbf{1 \leq M \leq 10^5}$
- All skill values given with be in the range [**0**, **20,000**]

- **1** ≤ **S** ≤ **N** for all queries
- All soldiers will have soldier **1** (Tywin) as their superior

---

Example

Input:

5 8

7 2 0 5 8

1 2

2 3

2 4

1 5

Q 1

Q 2

U 2 4

Q 1

Q 2

U 5 3

Q 1

Q 2

Output:

22

7

24

9

19

Day 4: PrimeSum

Abbas has been assigned the task to find the minimum number of 2-digit prime numbers which when added will be equal to a given number N.

The prototype of the function is as below -

int howManyPrimeNumsWillGet(int input1);

The function takes as input a number **input1**

1<= **input1**<=10^6

The function should return the minimum number of 2-digit prime numbers which when summed will result in **input1**.

If it is not possible to generate **input1** by adding 2-digit prime numbers, the function should return -1

Example1 –

If **input1**=28

Output: The function should return 2

Explanation:

11+17 = 28.

Number of 2-digit prime numbers which when added will produce 28 is **2**

So, the function returns **2**

Example2 –

If **input1**=14

Output: The function should return -1

Explanation:

14 cannot be generated by adding any 2-digit prime numbers.

So, the function returns **-1**

Example3 –

If **input1**=43

Output: The function should return 1

Explanation:

43 can be achieved by adding 11+13+19, BUT 43 itself is a 2-digit prime number, so the function should return **1**.

Example4 –

If **input1**=121

Output: The function should return 3

Explanation:

121 can be achieved by adding three 2-digit prime numbers, i.e. 11+13+97=121, so the function should return **3**.

Note: Any 2-digit prime number can also be used multiple times while trying to find the number of 2-digit prime numbers that add up to get **input1**.

Example5 –

If **input1**=33

Output: The function should return 3

Explanation:

33 can be achieved by adding three 2-digit prime numbers, i.e. 11+11+11=33, so the function should return **3**.

NOTE: The above examples are only few examples to help you understand the question. The actual test-case values will be different from these, so you must ensure to check the result for all possible cases.

Day 5: isPalindrome

Write a method to determine whether the input string is a Palindrome or not.

What is a Palindrome?

A palindrome is a string that spells the same from either directions, for example - abba, appa, amma, malayalam, nayan, deed, level, madam, rotator, reviver, stats, tenet, ...

The prototype of the method should be -

public static void isPalindrome(String input1);

where **input1** is the string to be checked for being palindrome.

If the input string is a palindrome, the method should assign **true** to the **output1** member.

If the input string is NOT a palindrome, the method should assign **false** to the **output1** member.

NOTE: The case of the letters in the string should not matter, i.e. Madam , MAdam , madAM , madam , MADAM , should all be considered a palindrome.

ASSUMPTIONS: Within the scope of this assessment, you can assume the following, and so you do not have to write code to handle the below conditions -

1. The passed input string will always be a single word and not a sentence

The passed input string will only contain alphabets.

Day 6: Playing World War II

Enigma: The military machine used by Nazi Germany to encipher messages before and during World War II. The code of Enigma was broken once by Polish Cipher Bureau in 1932 and then again, during the war, by the Codebreaker team at Bletchley Park, England. Because

of this, the Allied Powers were able to intercept a lot of messages transferred by German military and knew their battle plans. It is said that, breaking of Enigma shortened the length of that war by two years.

But, in the time of war, Allies were gravely concerned about the possibility of Germans finding out this security breach and resetting their communication system. They were very disciplined about using the information in order to hide this source of intelligence. Their precautions included leading fake invasions, superfluous search missions and attacking only after creating a proper cover story. Even at times, they considered sabotaging own resources during a battle which resulted in the death toll of hundreds of soldiers, civilians and massive loss on supplies; just to avoid any suspicion!

It seems like they played with probabilities leaving peoples' lives hanging on the decision.

Cruel, wasn't it? Can you do it?

There are **N** battles ahead. The probability that Allies will win the i^{th} battle is $\mathbf{P_i}$.

The enemies are calculating a value **PEB (Probability that Enigma has been broken)**. If the allies win a battle, the value of PEB increases; if the enemy wins the battle, the value decreases. If the value of PEB gets too high, the enemy will become suspicious and replace Enigma with something else.

PEB is calculated as an average from results of all previous battles using the following method.

```
total := 0

foreach Battle, b_i

    if Allies win

        total += probability of Allies losing in b_i

    else

        total -= probability of Allies winning in b_i

PEB = max(total / no. of battles, 0)
```

As the Allies already deciphered Enigma, they can choose to win or lose any battle they want. Consider the following scenario as example:

Battle No.	Probability, Allies Win	Probability, Allies lose (1 – winning probability)	Decision	PEB
1	0.7	0.3	Win	0.3
2	0.4	0.6	Win	(0.3+0.6)/2 = 0.45
3	0.8	0.2	Lose	(0.3+0.6-0.8)/3 = 0.033
4	0.3	0.7	Win	(0.3+0.6-0.8+0.7)/4 = 0.2
5	0.9	0.1	Lose	(0.3+0.6-0.8+0.7-0.9)/5 = -0.2 -> 0.0
6	0.6	0.4	Win	(0.3+0.6-0.8+0.7-0.9+0.4)/6 = 0.05

Some battles are more important than others; the i^{th} battle has an Impact Factor $\mathbf{F_i}$. As a war analyst, your job is to maximize the sum of Impact Factors of all the battles won. But the enemies must not know that Allies have deciphered Enigma, so you must keep PEB less than a certain **threshold, S**; throughout the whole war.

Input

- The first line of input contains an integer **T (1 ≤ T ≤ 100)** denoting the number of test cases. Each test case contains 4 lines:
- The first line contains one integer **N (1 ≤ N ≤ 100)** denoting the number of battles.
- The second line contains **N** space-separated real numbers $\mathbf{P_1,P_2,....,P_n}$ (probability of the Allies winning in i^{th} battle) $\mathbf{0.00 \le P_i \le 1.00}$.
- The third line contains **N** space-separated integers $\mathbf{F_1,F_2,...,F_n}$ (Impact factor of the i^{th} battle). Here $\mathbf{1 \le F_i \le 10000}$
- The fourth line contains a real number denoting threshold, **S (0.00 < S ≤ 1.00)**.

The real numbers contain at most 2 digits after the decimal point.

Output

For each test case, print case number and the maximum sum of impact factors.

Sample

Input

2

2

0.8 0.9

100 200

1

4

0.28 0.62 0.92 0.96

3 8 7 2

0.05

Output

Case 1: 300

Case 2: 9

Day 7: Weight of a hill pattern

Given,

the total levels in a hill pattern (input1),

the weight of the head level (input2), and

the weight increments of each subsequent level (input3),

you are expected to find the total weight of the hill pattern.

"Total levels" represents the number of rows in the pattern.

"Head level" represents the first row.

Weight of a level represents the value of each star (asterisk) in that row.

The hill patterns will always be of the below format, starting with 1 star at head level and increasing 1 star at each level till level N.

*

**

***

****

*****

******

. . .and so on till level N

Let us see a couple of examples.

Example1 -

Given,

the total levels in the hill pattern = 5 (i.e. with 5 rows)

the weight of the head level (first row) = 10

the weight increments of each subsequent level = 2

Then, The total weight of the hill pattern will be calculated as = 10 + (12+12) + (14+14+14) + (16+16+16+16) + (18+18+18+18+18) = 10 + 24 + 42 + 64 + 90 = 230

Example2 -

Given,

the total levels in the hill pattern = 4

the weight of the head level = 1

the weight increments of each subsequent level = 5

Then, Total weight of the hill pattern will be = 1 + (6+6) + (11+11+11) + (16+16+16+16) = 1 + 12 + 33 + 64 = 110

Day 8: maxMirror

We'll say that a "mirror" section in an array is a group of contiguous elements such that somewhere in the array, the same group appears in reverse order. For example, the largest mirror section in {1, 2, 3, 8, 9, 3, 2, 1} is length 3 (the {1, 2, 3} part). Return the size of the largest mirror section found in the given array.

maxMirror([1, 2, 3, 8, 9, 3, 2, 1]) → 3
maxMirror([1, 2, 1, 4]) → 3
maxMirror([7, 1, 2, 9, 7, 2, 1]) → 2

Column name from a given column number

Given a positive integer, return its corresponding column title as appear in an Excel sheet. MS Excel columns has a pattern like A, B, C, … ,Z, AA, AB, AC,…. ,AZ, BA, BB, … ZZ, AAA, AAB ….. etc. In other words, column 1 is named as “A”, column 2 as “B”, column 27 as “AA”.

Input:

The first line of each input consists of the test cases. And, the second line consists of a number N.

Output:

In each separate line print the corresponding column title as it appears in an Excel sheet.

Constraints:

$1 \leq T \leq 70$
$1 \leq N \leq 4294967295$

Example:

Input:

2
28
13

Output:

AB
M

Day 9: Special Keyboard

Imagine you have a special keyboard with the following keys:
Key 1: Prints 'A' on screen
Key 2: (Ctrl-A): Select screen
Key 3: (Ctrl-C): Copy selection to buffer
Key 4: (Ctrl-V): Print buffer on screen appending it
after what has already been printed.

If you can only press the keyboard for N times (with the above four keys), write a program to produce maximum numbers of A's. That is to say, the input parameter is N (No. of keys that you can press), the output is M (No. of As that you can produce).

Input:

The first line of input contains an integer T denoting the number of test cases.
The first line of each test case is N,N is the number of key.

Output:

Print maximum number of A's. Print -1, if N is greater than 75.

Constraints:

$1 \leq T \leq 50$
$1 \leq N \leq 75$

Example:

Input:
2
3
7

Output:
3
9

Explanation:

Input: N = 3
Output: 3
We can at most get 3 A's on screen by pressing
following key sequence.
A, A, A

Input: N = 7
Output: 9
We can at most get 9 A's on screen by pressing
following key sequence.
A, A, A, Ctrl A, Ctrl C, Ctrl V, Ctrl V

Day 10: MirrorEnds

Given a string, look for a mirror image (backwards) string at both the beginning and end of the given string. In other words, zero or more characters at the very begining of the given string, and at the very end of the string in reverse order (possibly overlapping). For example, the string "abXYZba" has the mirror end "ab".

mirrorEnds("abXYZba") → "ab"

mirrorEnds("abca") → "a"
mirrorEnds("aba") → "aba"

Day 11: ASSESSMENT 1

Day 12: Steps by Knight

Given a square chessboard of N x N size, the position of Knight and position of a target is given. We need to find out minimum steps a Knight will take to reach the target position.

Input:
The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case contains an integer n denoting the size of the square chessboard. The next line contains the X-Y coordinates of the knight. The next line contains the X-Y coordinates of the target.

Output:
Print the minimum steps the Knight will take to reach the target position.

Constraints:
1<=T<=100
1<=N<=20
1<=knight_pos,targer_pos<=N

Example:
Input:
2
6
4 5
1 1
20
5 7
15 20

Output:
3
9

Day 13: Element with left side smaller and right side greater

Given an unsorted array of size N. Find the first element in array such that all of its left elements are smaller and all right elements to it are greater than it.

Note: Left and right side elements can be equal to required element. And extreme elements

cannot be required element.

Input: The first line of input contains an integer T denoting the number of test cases. Then T test cases follow. Each test case consists of two lines. First line of each test case contains an Integer N denoting size of array and the second line contains N space separated array elements.

Output:
For each test case, in a new line print the required element. If no such element present in array then print -1.

Constraints:
1<=T<=100
3<=N<=10^6
1<=A[i]<=10^6

Example:
Input:
3
4
4 2 5 7
3
11 9 12
6
4 3 2 7 8 9

Output:
5
-1
7

Day 14: LUCKY FIVES

Everybody knows, that five is an extremely lucky number. It's even more lucky than seven!

For instance, five is widely used in Chinese culture. The famous Chinese Five Elements refer to Gold, Wood, Water, Fire and Earth, which are regarded as the basis of the whole world. In folklore, Chinese people hold the mutual restriction of the five elements: “wood restricting earth, earth restricting water, water restricting fire, fire restricting gold and gold restricting wood”. This mutual restriction principle is seriously stressed by the ancient Chinese people to show that everything in the world is mutually affected.

You see how important the number is? That's why this number will be under our consideration in this problem!

You are given a sequence $\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_N}$ of non-negative integers. Your task is to process **Q** queries of the following format: each query is described by two integers $\mathbf{L} \leq \mathbf{R}$ and asks to calculate the number of triples (**i**, **j**, **k**), such that $\mathbf{L} < \mathbf{i} < \mathbf{j} < \mathbf{k} < \mathbf{R}$ and $\mathbf{a_L} > \mathbf{a_i} < \mathbf{a_j} > \mathbf{a_k} < \mathbf{a_R}$.

Input

The first line of the input contains two integers **N** and **Q** denoting the size of the sequence and the number of the queries to process.

The second line contains **N** non-negative integers $\mathbf{a_1}$, $\mathbf{a_2}$, ..., $\mathbf{a_N}$.

Each of the next **Q** lines contains two integers **L** and **R** describing the corresponding query.

Output

For each query, output a single line containing one integers: the number of triples as described above.

Constraints

- $1 \leq \mathbf{N} \leq 2000$
- $1 \leq \mathbf{Q} \leq 100000$
- $0 \leq \mathbf{a_i} \leq 10^9$
- $1 \leq \mathbf{L} \leq \mathbf{R} \leq \mathbf{N}$ for each query in the input

Example

Input:

10 3

5 5 1 1 5 5 1 1 5 5

1 10

2 9

1 1

Output:

8

8

0

Explanation

The following triples(**i**, **j**, **k**) are valid for the first and the second queries:

- (3, 5, 7)
- (3, 5, 8)
- (3, 6, 7)
- (3, 6, 8)
- (4, 5, 7)
- (4, 5, 8)
- (4, 6, 7)
- (4, 6, 8

Day 15: Next Smallest Palindrome

Given a number, find the next smallest palindrome.

Input :
The first line is number of test cases, second is size of array, third line contains digits of number with spaces in between. The input is assumed to be an array. Every entry in array represents a digit in input number. Let the array be ‘num[ ]’ and size of array be ’n’. Size of the number can be upto 1000 digits.

Output:
In each separate line print the digits of palindrome with spaces in between.

Constraints:
1<=T<=100
1<=n<=1000
1<=a[i]<=9

Example:
Input:
1
11
9 4 1 8 7 9 7 8 3 2 2

Output:
9 4 1 8 8 0 8 8 1 4 9

Day 16: Boolean Parenthesization

Given a boolean expression with following symbols.

Symbols
'T' ---> true
'F' ---> false

And following operators filled between symbols

Operators
& ---> boolean AND
| ---> boolean OR
^ ---> boolean XOR

Count the number of ways we can parenthesize the expression so that the value of expression evaluates to true.

For Example:
The expression is "T | T & F ^ T", it evaluates true
in 4 ways ((T|T)&(F^T)), (T|(T&(F^T))), (((T|T)&F)^T)
and (T|((T&F)^T)).

Return No_of_ways Mod **1003**.

Input:
First line contains the test cases T. 1<=T<=500
Each test case have two lines
First is length of string N. 1<=N<=100
Second line is string S (boolean expression).
Output:
No of ways Mod 1003.

Example:
Input:
2
7
T|T&F^T
5
T^F|F

Output:
4
2

Day 17: LCS Revisited

The Longest Common Subsequence (LCS) problem is a well known problem in Computer Science. Every computer science students in Byteland knows this problem. Johnny does, too.

Recall that a subsequence of a string S is obtained by deleting some characters from S. Given two strings S and T, the LCS problem is the find the longest sequence that is a subsequence of both S and T.

Johnny has the habit of deriving harder problems from a familiar problem. This time, based on the LCS problem, he has thought up the following problem:

Given two strings S and T, how many distinct LCS of S and T are there?

Write a program to help Johnny solve this problem. Since the result may be very large, you only need to print the remainder of the result when dividing by 23102009.

Input

The first line contains t, the number of test cases (about 10). Then t test cases follow.

Each test case consists of two lines, the first line is the string S and the second line is the string T. You may assume that the strings consists of only lowercase characters and the length of the each string is at most 1000 characters.

Successive test cases at input are separated by a single blank line.

Output

For each test case, print a single line containing two numbers which are the length of the LCS and the remainder of the number of distinct LCS of S and T when dividing by 23102009.

Example

Input

2

acbd

acbd

vnvn

vn

Output

4 1

2 1

Output details

The only LCS in the first case is "acbd" and in the second case is "vn".

Day 18: Word Boggle

Given a dictionary, a method to do lookup in dictionary and a M x N board where every cell has one character. Find all possible words that can be formed by a sequence of adjacent characters. Note that we can move to any of 8 adjacent characters, but a word should not have multiple instances of same cell.

Example:

Input: dictionary[] = {"GEEKS", "FOR", "QUIZ", "GO"};

```
boggle[][]   = {{'G','I','Z'},
                {'U','E','K'},
                {'Q','S','E'}};
```

Output: Following words of dictionary are present

GEEKS, QUIZ

Input:

The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. Each test case contains an integer x denoting the no of words in the dictionary. Then in the next line are x space separated strings denoting the contents of the dictinory. In the next line are two integers N and M denoting the size of the boggle. The last line of each test case contains NxM space separated values of the boggle.

Output:

For each test case in a new line print the space separated sorted distinct words of the dictionary which could be formed from the boggle. If no word can be formed print -1.

Constraints:
1<=T<=10
1<=x<=10
1<=n,m<=7

Example:
Input:
1
4
GEEKS FOR QUIZ GO
3 3
G I Z U E K Q S E

Output:
GEEKS QUIZ

Day 19: Angle between hour and minute hand

Calculate the angle between hour hand and minute hand.

There can be two angles between hands, we need to print minimum of two. Also, we need to print **floor** of final result angle. For example, if the final angle is 10.61, we need to print 10.

Input:

The first line of input contains a single integer T denoting the number of test cases. Then T test cases follow. Each test case consists of one line containing two space separated numbers h and m where h is hour and m is minute.

Output:
Corresponding to each test case, print the angle b/w hour and min hand in a separate line.

Constraints:

$1 \leq T \leq 100$
$1 \leq h \leq 12$
$1 \leq m \leq 60$

Example:

Input
2

9 60
3 30

Output
90
75

Day 20: ASSESSMENT 2

Day 21: Pleasing the Chief

Chef's girlfriend, Chief, is displeased with him. This is not a good sign for their relationship, and as always, he has turned to you for help. He looks forward to take his girlfriend to shopping;
offering her to buy 'K' gifts. The shop that he is taking her to has 'N' unique items, that is, there is single quantity of each item (its a very special shop, you see!).

Chef has with him, K discount coupons that he wishes to use. The shop's policy only allows Chef to use one discount coupon on any one purchased item. That is why Chef is letting Chief buy K
gifts; he intends to use each coupon with him.

Chef of course cannot stop Chief from buying whatever she likes. Chief would select any K gifts, from among the N items that the shop has. She makes selection of any K out of N items, uniformly. Suppose,
there are 5 items and Chef lets Chief buy 3, then she selects any of the 3 out of 5 items with the probability 1/10.

After Chief selects the items she wishes to buy, Chef will have to pay the bill. Smart as he is, he would apply the K discount coupons on the K items (one on each) such that the discount he receives is as large
as possible.

Chef wants to prepare himself for this day. He knows beforehand the price of each item in the shop. Can you tell him what expected discount can he expect? You have to tell him the expected discount
in the amount of money he can expect to save.

Input format

The first line contains the number T, the number of test cases. In the following lines,
T test cases follow (without any newlines between them.)
Each case consists of only 3 lines.
The first line of each test case contains N and K, separated by a single space.
The second line contains N positive integers, the prices of the N items respectively, separated by a single space.

The third line contains K positive integers (between 1 and 100, inclusive) separated by a single space. They represent
the percentage of discount offered by the coupons, that Chef has, respectively.

Output format

For each test case, print the expected amount of money Chef should expect to save.
Output the result rounded to 3 digits after the decimal.

Constraints

$1 \leq T \leq 50$
$1 \leq N \leq 1000$
$1 \leq K \leq N$
$1 \leq \text{prices} \leq 10000$

Sample input

2

3 2

100 200 300

10 20

4 3

100 200 300 400

10 20 30

Sample output

66.667

175.000

Explanation

In the second case, Chief can make the following selections:

- (100, 200, 300), with probability 1/4. Chef can apply the coupons to at best save 140.
- (100, 200, 400), with probability 1/4. Chef can apply the coupons to at best save 170.
- (100, 300, 400), with probability 1/4. Chef can apply the coupons to at best save 190.

- (200, 300, 400), with probability 1/4. Chef can apply the coupons to at best save 200.

Thus Chef can save an expected (140 + 170 + 190 + 200) / 4 = 175.000

Day 22: Pairs with Positive Negative value

Given an array of distinct integers, print all the pairs having positive value and negative value of a number that exists in the array.

NOTE: If there is no such pair in the array, print "**0**".

Input:
The first line consists of an integer **T** i.e number of test cases. The first line of each test case consists of an integer **n**. The next line of each test case consists of **n** spaced integers.

Output:
Print all the required pairs in the increasing order of their absolute numbers.

Constraints:
1<=T<=100
1<=n<=1000
-1000<=a[i]<=1000

Example:
Input:
2
6
1 -3 2 3 6 -1
8
4 8 9 -4 1 -1 -8 -9

Output:
-1 1 -3 3
-1 1 -4 4 -8 8 -9 9

Day 23: DIGIT IN A RANGE

Find the number of times that a digit N occurs in all prime numbers from A to B (including A and B)?

Amar has been assigned the task to find the total number of times that a digit N occurs in all prime numbers from A to B, including in the numbers A and B.

The prototype of the function is as below -

int **getCountOfDigitNinPrimeRange**(int input1, int input2, int input3);

The input parameters of the function are as below -

input1 = N, the digit that is to be searched and counted

input2 = starting number of the range to be searched

input3 = ending number of the range to be searched

0<=input1<=9

0<=input2<=input3<=32000

The function should return the total number of times that the digit input1 occurs in all prime numbers occurring in the range from input2 to input3

Example1 –

input1 =3

input2 =11

input3 =37

Output: The function should return 4

Explanation:

The prime numbers occurring in the range from 11 to 37 are 11, 13, 17, 19, 23, 29, 31, 37

The digit 3 occurs a total of **4** times i.e. in 13, 23, 31 and 37.

So, the function returns **4**

Example2 –

input1 =1

input2 =11

input3 =37

Output: The function should return **6**

Explanation:

The prime numbers occurring in the range from 11 to 37 are 11, 13, 17, 19, 23, 29, 31, 37

The digit 1 occurs a total of **6** times i.e. in 11, 13, 17, 19, and 31.

So, the function returns **6**

Example3 –

input1 =6

input2 =11

input3 =37

Output: The function should return **0**

Explanation:

The prime numbers occurring in the range from 11 to 37 are 11, 13, 17, 19, 23, 29, 31, 37

The digit 6 occurs a total of **0** times in the above range.

So, the function returns **0**

NOTE: The above examples are only few examples to help you understand the question. The actual test-case values will be different from these, so you must ensure to check the result for all possible cases.

Day 24: Closest Palindrome

Given a number N. our task is to find the closest Palindrome number whose absolute difference with given number is minimum.

Input:
The first line of the input contains integer **T** denoting the number of test cases. Each test case contains a number N.

Output:
For each test case, the print the closest palindrome number.
Note: If the difference of two closest palindromes numbers is equal then we print smaller number as output.

Constraints:
1<=T<=1000
1<=n<=10^14

Input:
2
9
489

Output:
8
484

Explanation:

Test Case 1: closest palindrome number is 8.

Day 25: Maximum Tip Calculator

Rahul and Ankit are the only two waiters in Royal Restaurant. Today, the restaurant received N orders. The amount of tips may differ when handled by different waiters, if Rahul takes the ith order, he would be tipped A_i rupees and if Ankit takes this order, the tip would be $B_i$rupees.
In order to maximize the total tip value they decided to distribute the order among themselves. One order will be handled by one person only. Also, due to time constraints Rahul cannot take more than X orders and Ankit cannot take more than Y orders. It is guaranteed that X + Y is greater than or equal to N, which means that all the orders can be handled by either Rahul or Ankit. Find out the maximum possible amount of total tip money after processing all the orders.

Input:

- The first line contains one integer, number of test cases.
- The second line contains three integers N, X, Y.
- The third line contains N integers. The i^{th} integer represents A_i.
- The fourth line contains N integers. The i^{th} integer represents B_i.

Output:
Print a single integer representing the maximum tip money they would receive.
Constraints:
$1 \leq N \leq 105$
$1 \leq X, Y \leq N; X + Y \geq N$
$1 \leq Ai, Bi \leq 104$

Example:

Input:

2
5 3 3
1 2 3 4 5
5 4 3 2 1
8 4 4
1 4 3 2 7 5 9 6
1 2 3 6 5 4 9 8

Output:

21
43

Day 26: Sherlock and Smallest Window

Watson has infinite supply of K different type of objects denoted by numbers 1 to K. He makes an array A of N objects, A1, A2, ..., AN using these objects.

Now, he'll update this array Q times. In i^{th} update, he'll replace an existing object in array with a new object of different type from 1 to K i.e. Axi = yi.

After each update, he wants Sherlock to report the smallest size of a contiguous subarray which contains all types of objects i.e. smallest value of R - L + 1 such that number of distinct values in AL, AL + 1, ..., Ar is K. If no such subarray exists, Sherlock should output -1.

Input

First line contains N, K and Q denoting number of array elements, number of different types of objects and number of updates. Next line contains N space separated integers denoting the initial array A. ith of the next Q lines contains integers xi and yi, denoting the update parameters of the ith update.

Output

After each update, output an integer in one line denoting the smallest contiguous subarray size consisting of all types of objects. If no such subarray exists, output -1.

Constraints

$1 \leq N \leq 105$

$1 \leq K \leq 20$

$1 \leq Q \leq 105$

$1 \leq xi \leq N$

$1 \leq Ai, yi \leq K$

Example

Input:

5 3 3

1 1 1 1 1

2 3

5 2

3 2

Output:

-1

4

3

Explanation

After first update, the array is [1, 3, 1, 1, 1]. No valid subarray exists at this stage. After next update, the array is [1, 3, 1, 1, 2]. Here you can observe that contiguous subarray [A2, A3, ..., A5] consists of all types of objects. After next update, the array is [1, 3, 2, 1, 2]. Now, contiguous subarray [A1, A2, A3] is the smallest valid subarray.

Day 27 & 28: LARGE CITY DRUNK MAN

A city consists of n junctions that are connected by m bidirectional roads. The junctions are numbered from 1 to n. It is known that each junction has at least two roads incident on it. Also, one can walk from any junction to any other junction using these roads. Moreover, no pair of junctions is connected by more than one road.

A very drunk man has just walked out of a bar which is situated at junction s and wants to get to his home which is located at junction t. His current junction is s. In every minute the following happens:

- Step 1: He rationally thinks and chooses one of the roads incident on the current junction and walks along this road.
- Step 2: If he reaches the junction t, he immediately gets home and goes to sleep.

- Step 3: Otherwise, influenced by the desire for adventure, he picks some random road incident on the current junction and walks along this road. The only limitation is that he never picks the road he used to get to this junction at the beginning of the current minute (ie. in Step 1 of this minute).
- Step 4: Now, if he has reached junction t, he sees his home and goes there directly and sleeps.
- Step 5: If not, then we go to the next minute, and this repeats.

Given a description of the city and indices of junctions s and t you have to compute whether the drunk man will surely get home in a finite number of minutes, if he makes optimal choices in all the Step 1's. If so, what is the minimum number of minutes x, such that the man can get home in no more than x minutes whatever his random choices at step 3 of each minute are. That is, what is the minimum x, such that there is a strategy which the man can follow in his Step 1 choices, so that no matter what adversarial choice he makes in the Step 3's, he is guaranteed to reach his home within x minutes?

Input

- The first line of the input contains a single integer T — the number of test cases in this input. Then follow T descriptions of individual test cases.
- The first line of the input contains four integers n, m, s and t — the number of intersections and the number of roads in the city, the index of the intersection where the bar is located and the index of the intersection where the drunk man's house is, respectively.
- The ith of the following m lines contains two integers ui and vi — indices of the junctions connected by the ith road.

Output

- If there is no strategy for him to reach home definitely, output -1 in a new line.
- Otherwise, print how many minutes the drunk man needs to get home in the worst possible scenario if he always behaves optimally in the Step 1 of every minute.

Constraints

- $1 \leq T \leq 100000$
- $3 \leq n, m \leq 300000$
- $1 \leq s, t \leq n$
- $s \neq t$

- $1 \leq ui, vi \leq n$
- $ui \neq vi$
- It is guaranteed that it is possible to get from any junction to any other junction moving along the roads, each junction has at least two incident roads and no two roads connect the same pair of junctions.
- It is guaranteed that both the total number of intersections and the total number of roads over all test cases in one input file won't exceed 1000000

Example

Input:

2

4 5 1 2

1 3

1 4

3 4

2 4

2 3

6 9 1 6

1 2

1 3

2 3

2 4

2 5

3 4

3 5

4 6

5 6

Output:

2

-1

Explanation

Testcase 1: Consider the strategy where the drunk man first moves to junction 3 in Step 1 of the first minute. Then in Step 3 of the first minute, he randomly picks the road to junction 2 or the road to junction 4. In case he had picked the road to junction 2, he reaches home in just 1 minute. If not, he ends up at junction 4 at the end of the first minute. Then, in Step 1 of the second minute, he picks the road between 4 and 2, and goes to junction 2, thus reaching his home in the second minute. Thus, no matter what random choices he makes in the Step 3's, he is guaranteed to reach his house within the first two minutes.

Testcase 2: You can show that in the worst case, the drunk man will could be stuck between among the junctions {1, 2, 3, 4, 5} forever.

Day 29: Find all four sum numbers

Given an array A of size N, find all combination of four elements in the array whose sum is equal to a given value K. For example, if the given array is {10, 2, 3, 4, 5, 9, 7, 8} and K = 23, one of the quadruple is “3 5 7 8” (3 + 5 + 7 + 8 = 23).

Input:
The first line of input contains an integer T denoting the no of test cases. Then T test cases follow. Each test case contains two lines. The first line of input contains two integers N and K. Then in the next line are N space separated values of the array.

Output:
For each test case in a new line print all the quadruples present in the array separated by space which sums up to value of K. Each quadruple is unique which are separated by a delimeter "$" and are in increasing order.

Constraints:
1<=T<=100
1<=N<=100
-1000<=K<=1000
-100<=A[]<=100

Example:
Input:
2
5 3
0 0 2 1 1
7 23
10 2 3 4 5 7 8

Output:

0 0 1 2 $
2 3 8 10 $2 4 7 10 $3 5 7 8 $