1. How Many X's?

Given an integer X within the range of 0 to 9, and given two positive integers as upper and lower bounds respectively, find the number of times X occurs as a digit in an integer within the range, excluding the bounds. Print the frequency of occurrence as output.

Note : Save the file / class as “countXInRange” with appropriate file extension.

Input:

The first line of input is an integer T, denoting the number of test cases. For each test case, there are two lines of input, first consisting of the integer X, whose occurrence has to be counted. Second, the lower and upper bound, L and U which are positive integers, on the same line separated by a single space, respectively.

Output:

For each test case, there is only one line of output, the count of the occurrence of X as a digit in the numbers lying between the lower and upper bound, excluding them.

Constraints:

1<=T<=100

0<=X<=9

0<L<U<=10^5

Example:

Input:

2

3

100 250

0

20 21

Output:

35

0

Explanation:

In the first test case, the occurrence of 3 in the numbers starting from 101 to 249 is counted and comes out to be 35.

Similarly, for all the other test cases, the occurrence of the given number X is printed as output.

Input:

3

2

10000 12345

9

899 1000

1

1100 1345

Output:

1120

120

398

2. Find the Minimum difference pair

Given an unsorted array, find the minimum difference between any pair in given array.

Note : Save the file / class as “minDifferencePair” with appropriate file extension.

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, the size of array. Second line of the test case is the Array.

Output:

Print the minimum difference between any two pairs.

Constraints:

1 <= T <= 30

1 < N <= 100

1 <= arr[i] <= 100000

Example:

Input:

2

5

2 4 5 7 9

10

87 32 99 75 56 43 21 10 68 49

Output:

1

6

3. Find the Maximum money

Given street of houses (a row of houses), each house having some amount of money kept inside; now there is a thief who is going to steal this money but he has a constraint/rule that he cannot steal/rob two adjacent houses. Find the maximum money he can rob.

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 and money.

Output:

Print maximum money he can rob.

Note : Save the file / class as “maxMoney” with appropriate file extension.

Constraints:

1 ≤ T ≤ 100

1 ≤ money ≤ 100

1 ≤ N ≤ 1000

Example:

Input:

2

5 10

2 12

Output:

30

12

4. Rotate Array

Given an unsorted array arr[] of size N, rotate it by D elements (anti-clockwise).

Note : Save the file / class as “rotateArrayNTimes” with appropriate file extension.

Input:

The first line of the input contains T denoting the number of testcases. First line of eacg test case contains two space separated elements, N denoting the size of the array and an integer D denoting the number size of the rotation. Subsequent line will be the N space separated array elements.

Output:

For each testcase, in a new line, output the rotated array.

Constraints:

1 <= T <= 200

1 <= N <= 107

1 <= D <= N

0 <= arr[i] <= 105

Example:

Input:

2

5 2

1 2 3 4 5

10 3

2 4 6 8 10 12 14 16 18 20

Output:

3 4 5 1 2

8 10 12 14 16 18 20 2 4 6

Explanation :

Testcase 1: 1 2 3 4 5 when rotated by 2 elements, it becomes 3 4 5 1 2

5. Twice counter

Given an array of n words. Some words are repeated twice, we need count such words.

Note : Save the file / class as “wordTwiceCounter” with appropriate file extension.

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 number of words in the string. The next line contains n space separated words forming the string.

Output:

Print the count of the words which are repeated twice in the string.

Constraints:

1<=T<=105

1<=no of words<=105

1<=length of each word<=105

Example:

Input:

2

10

hate love peace love peace hate love peace love peace

8

Tom Jerry Thomas Tom Jerry Courage Tom Courage

Output:

1

2

6. Count total set/one bits

You are given a number N. Find the total count of set bits (number of 1s when the number is represented in binary) form for all numbers from 1 to N(both inclusive).

Note : Save the file / class as “countSetBits” with appropriate file extension.

Input:

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

Output:

For each testcase, in a new line, print the total count of all bits.

Constraints:

1 ≤ T ≤ 100

1 ≤ N ≤ 103

Example:

Input:

2

4

17

Output:

5

35

Explanation:

Testcase1:

An easy way to look at it is to consider the number, n = 4:

0 0 0 = 0

0 0 1 = 1

0 1 0 = 1

0 1 1 = 2

1 0 0 = 1

Therefore , the total number of bits is 5.

7. Remove Characters in a String

Given two strings s1 and s2, remove those characters from first string which are present in second string. Both the strings are different and contain only lowercase characters.

Note : Save the file / class as “removeStringChars” with appropriate file extension.

Input:

The first line of input contains an integer T denoting the number of test cases.

The first line of each test case is s1,s1 is first string.

The second line of each test case contains s2,s2 is second string.

Output:

Print the modified string(s1). For each test case, print the output in a new line.

Constraints:

1 ≤ T ≤ 15

1 ≤ s2 < s1 ≤ 50

Example:

Input:

2

itvaccodingteam

caw

removeccharaterfrom

string

Output:

itvodingtem

emovecchaaefom

8. Generate Binary Numbers

Given a number N. The task is to generate and print all binary numbers with decimal values from 1 to N.

Note : Save the file / class as “genBinaryNos” with appropriate file extension.

Input:

The first line of input contains an integer T denoting the number of test cases. There will be a single line for each testcase which contains N.

Output:

Print all binary numbers with decimal values from 1 to N in a single line.

Constraints:

1 ≤ T ≤ 106

1 ≤ N ≤ 106

Example:

Input:

2

2

5

Output:

1 10

1 10 11 100 101

Explanation:

Testcase 1: Binary numbers from 1 to 2 are 1 and 10.

9. Keypad typing

You are given a string S of alphabet characters and the task is to find its matching decimal representation as on a mobile phone's numeric keypad. Output the decimal representation corresponding to the string. For ex: if you are given “amazon” then its corresponding decimal representation will be 262966.

Note : Save the file / class as “stringKeyMapping” with appropriate file extension.

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 a single line containing a string.

Output:

For each test case, print in a new line, the corresponding decimal representation of the given string.

Reference



Constraints:

1 ≤ T ≤ 100

1 ≤ length of String ≤ 100

Example:

Input

2

itvaccoding

vacquiz

Output

48822263464

8227849

10. Greater on right side

You are given an array A of size N. Replace every element with the next greatest element (greatest element on its right side) in the array. Also, if there is no element next to the last element, replace it with -1.

Note : Save the file / class as “greatOnRight” with appropriate file extension.

Input:

The first line of input contains an integer T denoting the number of test cases. T testcases follow. Each test-case contains two lines of input. The first line is N, the size of tha array. The second line contains N space separated integers.

Output:

For each test-case, print the modified array.

Constraints:

1 <= T <= 50

1 <= N <= 100

1 <= Ai <= 1000

Example:

Input:

2

6

16 17 4 3 5 2

4

2 3 1 9

Output:

17 5 5 5 2 -1

9 9 9 -1

Explanation:

Testcase1: For 16 the greatest element on its right is 17. For 17 it's 5. For 4 it's 5. For 3 it's 5. For 5 it's 2. For 2 it's -1(no element to its right). So the answer is 17 5 5 5 2 -1

11. Addition of sub-matrix

Given a matrix C of size N x M. You are given position of sub-matrix as X1, Y1 and X2, Y2 inside the matrix. Find the sum of all elements inside that sub-matrix.

Note : Save the file / class as “addSubMatrix” with appropriate file extension.

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 and m,n is the number of rows and m is the number of columns. The second line of each test case contains C[N][M]. The third line contains four value of X1, Y1, X2, Y2. X1, Y1 is the top left cell and X2, Y2 is the bottom right cell.

Output:

Print the sum of all elements inside that sub-matrix.

Constraints:

1 ≤ T ≤ 15

1 ≤ N, M ≤ 103

1 ≤ C[N][M] ≤ 106

1 <= X1, Y1, X2, Y2 <= M

Example:

Input:

2

5 6

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30

3 4 4 5

3 3

9 8 7 4 2 1 6 5 3

1 2 3 3

Output:

78

26

Explanation:

Testcase 2: Sum from cell starting at position (1, 2) (1-based indexing) and ending at (3, 3) is 26.

12. Geek and Coffee Shop

Geek loves to drink cold coffee after coding for long hours. One fine day a Geek went to his favorite coffee shop in the town and asked for a cup of cold coffee. The shopkeeper told him that he is their lucky customer and had an offer for the Geek. The offer was that for an amount of N they will fill Geek's cup with N units of cold coffee and after consuming initial N units of coffee the shopkeeper will again refill his cup with half the amount of coffee that Geek consumed in previous refill, and will keep on refilling his cup by half of the previous cycle till the amount to refill becomes nil i.e. 0 (Assume Geek can consume infinite amount of coffee and shop also has infinite amount coffee). Now Geek is curious to know that how many units of coffee will Geek get for Mth refill. Being Geek's friend help him out with his problem.

Note : Save the file / class as “geeksCoffee” with appropriate file extension.

Input:

The first line of the input contains an integer T, denoting the number of test cases. The T test case follow. The only line of each test case contains two space separated integers N and M respectively.

Output:

For each test case output a single integer on a new line denoting the required answer.

Constraints:

1<=T<=104

1<=N<=109

1<=M<=103

Example:

Input:

2

100 4

10 3

Output:

12

2

Explanation:

TestCase 1:

For the 4th refill geek will get 12 units of the coffee.

1st Fill: Geek will get 100 units of Coffee

2nd Fill: 100/2 = 50 units

3rd Fill: 50/2 = 25 units

4th Fill: 25/2 = 12 units

13. Maximum product of two numbers

Given an array with all elements greater than or equal to zero. Return the maximum product of two numbers possible.

Note : Save the file / class as “maxProductInArray” with appropriate file extension.

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, size of array. The second line of each test case contains array elements.

Output:

Print the maximum product of two numbers possible.

Constraints:

1 ≤ T ≤ 100

2 ≤ N ≤ 107

0 ≤ A[i] ≤ 104

Example:

Input:

2

5

1 100 42 4 23

3

20 30 40

Output:

4200

1200

Explanation:

Testcase 1: Two maximum numbers are 100 and 42 and their product is 4200.

14. Decode the pattern

Given a pattern as below and an integer n your task is to decode it and print nth row of it. The pattern follows as :

11

121

1331

14641

Note : Save the file / class as “oneOnePattern” with appropriate file extension.

Input:

The first line of input is the number of test cases . Then T test cases follow . The first line of each test case is an integer N.

Output:

For each test case print the required nth row of the pattern.

Constraints:

1<=T<=20

1<=N<=20

Example:

Input:

2

2

4

Output:

11

121

11

121

1331

14641

15. Implement strstr manually

Your task is to implement the function strstr. The function takes two strings as arguments (s,x) and locates the occurrence of the string x in the string s. The function returns and integer denoting the first occurrence of the string x in s.

Note : Save the file / class as “manualSubString” with appropriate file extension.

Input Format:

The first line of input contains an integer T denoting the no of test cases . Then T test cases follow. Each test case has 2 lines of input, first list the search string s and second line the occurrence of string x

Output Format:

For each test case, in a new line, output will be an integer denoting the first occurrence of the x in the string s. Return -1 if no match found.  
NOTE: If there is a match on the first element answer should be 0 and not 1 given that array index starts at 0

Constraints:

1 <= T <= 100

1<= |s|,|x| <= 1000

Example:

Input

2

ItVACForSuccess

Fr

ITVACForsuccess

For

Output

-1

5

Explanation:

Testcase 1: Fr is not present in the string ItVACForSuccess as substring.

Testcase 2: For is present as substring in ITVACForsuccess from index 5 given that array index starts at 0.

16. Binary String

Given a binary string S. The task is to count the number of substrings that start and end with 1. For example, if the input string is “00100101”, then there are three substrings “1001”, “100101” and “101”.

Note : Save the file / class as “binaryStartAndEnd” with appropriate file extension.

Input:

The first line of input contains an integer T denoting the number of test cases. Each test case consist of an integer 'n' denoting the string length and next line is followed by a binary string.

Output:

For each testcase, in a new line, print the number of substring starting and ending with 1 in a separate line.

Constraints:

1 ≤ T ≤ 40

1 ≤ |S| ≤ 1000

Example:

Input:

2

4

1111

5

01101

Output:

6

3

Example:

Testcase 1: There are 6 substrings from the given string. They are 11, 11, 11, 111, 111, 1111.

Testcase 2: There 3 substrings from the given string. They are 11, 101, 1101.

17. Reverse each word in a given string

Given a String of length N reverse each word in it. Words are separated by spaces. Dot character should not be reversed and should retain its position.

Note : Save the file / class as “reverseEachWord” with appropriate file extension.

Input:

The first line contains T denoting the number of testcases. Then follows description of testcases. Each case contains a string containing dots, spaces and characters.

Output:

For each test case, output a String in single line containing the reversed words of the given String.

Constraints:

1<=T<=10

1<=Length of String<=2000

Example:

Input:

2

i like this program very much

hello

Output:

i ekil siht margorp yrev hcum

olleh

18. Given a String of length N capitalize the first letter of each word in the sentence

Input:

The first line contains T denoting the number of testcases. Then follows description of testcases. Each case contains a string containing dots and characters.

Note : Save the file / class as “capitalizeFirstLetter” with appropriate file extension.

Output:

For each test case, output a String in single line containing the words of the given String with first letter as upper case (capitals)

Constraints:

1<=T<=10

1<=Length of String<=2000

Example:

Input:

2

i like this. program very much

….

Output:

I Like This. Program Very Much

…

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Sorting Elements of an Array by Frequency

Given an array A[] of integers, sort the array according to frequency of elements. That is elements that have higher frequency come first. If frequencies of two elements are same, then smaller number comes first.

Input:

The first line of input contains an integer T denoting the number of test cases. The description of T test cases follows. The first line of each test case contains a single integer N denoting the size of array. The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array.

Output:

For each testcase, in a new line, print each sorted array in a seperate line. For each array its numbers should be seperated by space.

Constraints:

1 ≤ T ≤ 70

30 ≤ N ≤ 130

1 ≤ Ai ≤ 60

Example:

Input:

2

5

5 5 4 6 4

5

9 9 9 2 5

Output:

4 4 5 5 6

9 9 9 2 5

Explanation:

Testcase1: The highest frequency here is 2. Both 5 and 4 have that frequency. Now since the frequencies are same then smaller element comes first. So 4 4 comes first then comes 5 5. Finally comes 6.

The output is 4 4 5 5 6.

Testcase2: The highest frequency here is 3. The element 9 has the highest frequency. So 9 9 9 comes first. Now both 2 and 5 have same frequency. So we print smaller element first.

The output is 9 9 9 2 5.

2. Extract the Number from the String

Bastin once had trouble finding the numbers in a string. The numbers are distributed in a string across various test cases.

There are various numbers in each test case you need to find the number in each test case. Each test case has various numbers in sequence. You need to find only those numbers which do not contain 9. For eg, if the string contains "hello this is alpha 5051 and 9475".You will extract 5051 and not 9475. You need only those numbers which are consecutive and you need to help him find the numbers.

Note: Use long long for storing the numbers from the string.

Input:

The first line consists of T test cases and next T lines contain a string.

Output:

For each string output the number stored in that string if various numbers are there print the largest one. If a string has no numbers print -1.

Constraints:

1<=T<=100

1<=|S|<=10000

Example:

Input:

1

This is alpha 5057 and 97

Output:

5057

3. Valid Parenthesis String

Given a string containing only three types of characters: '(', ')' and '\*', write a function to check whether this string is valid. We define the validity of a string by these rules:

Any left parenthesis '(' must have a corresponding right parenthesis ')'.

Any right parenthesis ')' must have a corresponding left parenthesis '('.

Left parenthesis '(' must go before the corresponding right parenthesis ')'.

'\*' could be treated as a single right parenthesis ')' or a single left parenthesis '(' or an empty string.

An empty string is also valid.

Example 1:

Input: "()"

Output: True

Example 2:

Input: "(\*)"

Output: True

Example 3:

Input: "(\*))"

Output: True

Note:

The string size will be in the range [1, 100].

4. Time Conversion

Given a time in 12[-hour AM/PM format](https://en.wikipedia.org/wiki/12-hour_clock), convert it to military (24-hour) time.

Note: Midnight is 12:00:00AM on a 12-hour clock, and 00:00:00 on a 24-hour clock. Noon is 12:00:00PM on a 12-hour clock, and 12:00:00 on a 24-hour clock.

Input Format

A single string s containing a time in 12-hour clock format (i.e.: hh:mm:ssAM or hh:mm:ssPM ), where 01<= hh <= 12 and 00 <= mm, ss <= 59.

Constraints

All input times are valid

Output Format

Convert and print the given time in 24-hour format, where 00 <= hh <= 23.

Sample Input 0

07:05:45PM

Sample Output 0

19:05:45

5. You have been asked to help study the population of birds migrating across the continent. Each type of bird you are interested in will be identified by an integer value. Each time a particular kind of bird is spotted, its id number will be added to your array of sightings. You would like to be able to find out which type of bird is most common given a list of sightings. Your task is to print the type number of that bird and if two or more types of birds are equally common, choose the type with the smallest ID number.

For example, assume your bird sightings are of types arr=[1,1,2,2,3]. There are two each of types 1 and 2 , and one sighting of type 3. Pick the lower of the two types seen twice: type 1.

**Function Description**

Complete the *migratoryBirds* function in the editor below. It should return the lowest type number of the most frequently sighted bird.

migratoryBirds has the following parameter(s):

* *arr*: an array of integers representing types of birds sighted

**Input Format**

The first line contains an integer denoting n, the number of birds sighted and reported in the array arr.   
The second line describes arr as n space-separated integers representing the type numbers of each bird sighted.

**Constraints**

* 5 <= n <= 2\*10^5
* It is guaranteed that each type is 1, 2, 3, 4, or 5.

**Output Format**

Print the type number of the most common bird; if two or more types of birds are equally common, choose the type with the smallest ID number.

**Sample Input 0**

6

1 4 4 4 5 3

**Sample Output 0**

4

**Explanation 0**

The different types of birds occur in the following frequencies:

* Type 1: 1 bird
* Type 2: 0 birds
* Type 3: 1 bird
* Type 4: 3 birds
* Type 5: 1 bird

The type number that occurs at the highest frequency is type , so we print  as our answer.

**Sample Input 1**

11

1 2 3 4 5 4 3 2 1 3 4

**Sample Output 1**

3

**Explanation 1**

The different types of birds occur in the following frequencies:

* Type 1: 2 birds
* Type 2: 2 birds
* Type 3: 3 birds
* Type 4: 3 birds
* Type 5: 1 bird

Two types have a frequency of 3, and the lower of those is type 4.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Medium Level – Set 2**

1. Sorting Elements of an Array by Frequency

Given an array A[] of integers, sort the array according to frequency of elements. That is elements that have higher frequency come first. If frequencies of two elements are same, then smaller number comes first.

Note : The file/class should be named as "SortNumArrayByFreq" with appropriate file extension.

Input:

The first line of input contains an integer T denoting the number of test cases. The description of T test cases follows. The first line of each test case contains a single integer N denoting the size of array. The second line contains N space-separated integers A1, A2, ..., AN denoting the elements of the array.

Output:

For each test-case, in a new line, print each sorted array in a separate line. For each array its numbers should be separated by space.

Constraints:

1 <= T <= 70

30 <= N <= 130

1 <= Ai <= 60

Example:

Input 1:

2

5

5 5 4 6 4

5

9 9 9 2 5

Output 1:

4 4 5 5 6

9 9 9 2 5

Explanation:

Testcase1: The highest frequency here is 2. Both 5 and 4 have that frequency. Now since the frequencies are same then smaller element comes first. So 4 4 comes first then comes 5 5. Finally comes 6.

The output is 4 4 5 5 6.

Input 1:

3

7

3 4 9 5 4 2 3

7

5 4 5 2 4 3 2

15

9 9 2 5 12 5 4 2 3 3 10 4 9 11 15

Output 1:

2 3 3 4 4 5 9

2 2 3 4 4 5 5

2 2 3 3 4 4 5 5 9 9 9 10 11 12 15

2. Extract the Number from the String

Benny once had trouble finding the numbers in a string. The numbers are distributed in a string across various test cases.

There are various numbers in each test case you need to find the number in each test case. Each test case has various numbers in sequence. You need to find only those numbers which do not contain 9. For eg, if the string contains "hello this is alpha 5051 and 9475".You will extract 5051 and not 9475. You need only those numbers which are consecutive and you need to help him find the numbers.

Note: Use long long for storing the numbers from the string.

Note : The file/class should be named as "ExtractNumFromString" with appropriate file extension.

Input:

The first line consists of T test cases and next T lines contain a string.

Length of each string S may be between 1 to 1000.

Output:

For each string output the number stored in that string if various numbers are there print the largest one. If a string has no numbers print -1.

Constraints:

1<=T<=100

1<=|S|<=1000

Example:

Input 1:

2

This is alpha 5057 and 97

GSLV F11 was launched in December 19, 2018

Output 1:

5057 97

11 19 2018

Input 2:

3

In 2nd innings India scored 436 runs with 3 more wickets

In winter temperature falls below -4 degree Celsius

Absolute zero is defined as 0 K on the Kelvin scale and –273.15 degrees Celsius on the Celsius scale.

Output 2:

2 436 3

-4

0 -273.15

3. Valid Parenthesis String

Given a string containing only three types of characters: '(', ')' and '\*', write a function to check whether this string is valid. We define the validity of a string by these rules:

Any left parenthesis '(' must have a corresponding right parenthesis ')'.

Any right parenthesis ')' must have a corresponding left parenthesis '('.

Left parenthesis '(' must go before the corresponding right parenthesis ')'.

'\*' could be treated as a single right parenthesis ')' or a single left parenthesis '(' or an empty string.

An empty string is also valid.

Note : The file/class should be named as "ValidParenthesis" with appropriate file extension.

Note:

The string size will be in the range [1, 100].

Example:

Input 1 :

3

()

(\*)

)(\*)

Output 1 :

True

True

False

Input 2 :

3

(\*))

(\*()(

(())

Output 2 :

True

False

True

4. Time Conversion

Given a time in 12-hour AM/PM format, convert it to military (24-hour) time.

Note: Midnight is 12:00:00AM on a 12-hour clock, and 00:00:00 on a 24-hour clock. Noon is 12:00:00PM on a 12-hour clock, and 12:00:00 on a 24-hour clock.

Note : The file/class should be named as "TimeConversion" with appropriate file extension.

Input Format

A single string s containing a time in 12-hour clock format (i.e.: hh:mm:ssAM or hh:mm:ssPM ), where 01<= hh <= 12 and 00 <= mm, ss <= 59.

Constraints

All input times are valid

Output Format

Convert and print the given time in 24-hour format, where 00 <= hh <= 23.

Sample Input 1 :

07:05:45PM

Sample Output 1 :

19:05:45

Sample Input 2 :

12:05:15PM

Sample Output 2 :

12:05:15

Sample Input 3 :

12:15:00AM

Sample Output 3 :

00:15:00

Sample Input 4 :

08:30:00AM

Sample Output 4 :

08:30:00

5. You have been asked to help study the population of birds migrating across the continent.

Each type of bird you are interested in will be identified by an integer value.

Each time a particular kind of bird is spotted, its id number will be added to your array of sightings.

You would like to be able to find out which type of bird is most common given a list of sightings.

Your task is to print the type number of that bird and if two or more types of birds are equally common, choose the type with the smallest ID number.

For example, assume your bird sightings are of types arr=[1,1,2,2,3]. There are two each of types 1 and 2 , and one sighting of type 3. Pick the lower of the two types seen twice: type 1.

Note : The file/class should be named as "TimeConversion" with appropriate file extension.

Input Format

The first line contains an integer denoting n, the number of birds sighted and reported in the array arr.

The second line describes arr as n space-separated integers representing the type numbers of each bird sighted.

Constraints

5 <= n <= 2\*10^5

It is guaranteed that each type is 1, 2, 3, 4, or 5.

Output Format

Print the type number of the most common bird; if two or more types of birds are equally common, choose the type with the smallest ID number.

Sample Input 0

6

1 4 4 4 5 3

Sample Output 0

4

Explanation 0

The different types of birds occur in the following frequencies:

Type 1: 1 bird

Type 2: 0 birds

Type 3: 1 bird

Type 4: 3 birds

Type 5: 1 bird

The type number that occurs at the highest frequency is type 4, so we print 4 as our answer.

Sample Input 1

11

1 2 3 4 5 4 3 2 1 3 4

Sample Output 1

3

Explanation 1

The different types of birds occur in the following frequencies:

• Type 1: 2 birds

• Type 2: 2 birds

• Type 3: 3 birds

• Type 4: 3 birds

• Type 5: 1 bird

Two types have a frequency of 3, and the lower of those is type 3.

-------------------------------------------------------------------

1. Write a program to find longest binary gap in a given number.

A binary gap within a positive integer N is any maximal sequence of consecutive zeros that is surrounded by ones at both ends in the binary representation of N.

Note : The file/class should be named as "LongestBinaryGap" with appropriate file extension.

For example,

Number 9 has binary representation 1001 and contains a binary gap of length 2.

The number 529 has binary representation 1000010001 and contains two binary gaps: one of length 4 and one of length 3.

The number 20 has binary representation 10100 and contains one binary gap of length 1.

The number 15 has binary representation 1111 and has no binary gaps.

The number 32 has binary representation 100000 and has no binary gaps.

For example, given N = 1041 the function should return 5, because N has binary representation 10000010001 and so its longest binary gap is of length 5. Given N = 32 the function should return 0, because N has binary representation '100000' and thus no binary gaps.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [1..2,147,483,647].

Input Format :

Input contains a single line which consists of a single Integer.

Output Format :

Output consists of a single integer.

Test Case 1 :

Input :

125

Output :

1

Test Case 2 :

Input :

32

Output :

0

Test Case 3 :

Input :

714

Output :

2

Test Case 4 :

Input :

305

Output :

3

2. Write a program to calculate profit from a set of share prices.

An array A consisting of N integers is given. It contains daily prices of a stock share for a period of N consecutive days. If a single share was bought on day P and sold on day Q, where 0 <= P <= Q < N, then the profit of such transaction is equal to A[Q] − A[P], provided that A[Q] >= A[P].

Otherwise, the transaction brings loss of A[P] − A[Q].

For example, consider the following array A consisting of six elements such that:

A[0] = 23171 A[1] = 21011 A[2] = 21123 A[3] = 21366 A[4] = 21013 A[5] = 21367

If a share was bought on day 0 and sold on day 2, a loss of 2048 would occur because A[2] − A[0] = 21123 − 23171 = −2048.

If a share was bought on day 4 and sold on day 5, a profit of 354 would occur because A[5] − A[4] = 21367 − 21013 = 354.

Maximum possible profit was 356. It would occur if a share was bought on day 1 and sold on day 5.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [0..400,000];

each element of array A is an integer within the range [0..200,000].

Note : The file/class should be named as "ShareMarketProfit" with appropriate file extension.

Input Format :

First line of input consists of a single integer 'N' which is the size of the array.

Next 'N' lines consists of 'N' integers which are part of array A.

Output Format :

Output consists of a single integer which is the maximum profit that the user can earn.

Test Input 1 :

6

23187

92727

231

4563

123131

1311

Output:

122900

Test Input 2 :

4

200

200

200

200

Output:

0

Test Input 3 :

6

123

754

1543

2413

8575

1231

Output:

8452

Test Input 4 :

6

500

1500

3000

2800

5000

4000

Output:

3500

3. Write a program to find the number of semi-prime numbers within a set of ranges.

A prime is a positive integer X that has exactly two distinct divisors: 1 and X. The first few prime integers are 2, 3, 5, 7, 11 and 13.

A semi-prime is a natural number that is the product of two (not necessarily distinct) prime numbers. The first few semi-primes are 4, 6, 9, 10, 14, 15, 21, 22, 25, 26.

You are given two non-empty arrays P and Q, each consisting of M integers. These arrays represent queries about the number of semi-primes within specified ranges.

Query K requires you to find the number of semi-primes within the range (P[K], Q[K]), where 1 <= P[K] <= Q[K] <= N.

For example, consider an integer N = 26 and arrays P, Q such that:

P[0] = 1 Q[0] = 26 P[1] = 4 Q[1] = 10 P[2] = 16 Q[2] = 20

The number of semi-primes within each of these ranges is as follows:

(1, 26) is 10,

(4, 10) is 4,

(16, 20) is 0.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [1..50,000];

each element of arrays P, Q is an integer within the range [1..1000];

P[i] <= Q[i].

Note : The file/class should be named as "SemiPrimeArray" with appropriate file extension.

Input Format :

First line of input consists of a single integer 'N' which is the size of the array.

Next 2\*N lines consists of N integers which are part of array P and N integers which are part of array Q.

Output Format :

Output consists of a single integer which is the maximum profit that the user can earn.

Test case 1:

4

1

12

34

23

12

23

67

71

Output :

4 4 12 16

Test case 2:

3

1

5

20

12

14

23

Output :

4 4 2

Test case 3:

4

6

8

12

26

18

81

70

38

Output :

5 4 24 5

Test case 4:

4

4

2

5

3

13

7

14

9

Output :

4 2 4 3

4. Write a program to find the dominator element of an array.

An array A consisting of N integers is given. The dominator of array A is the value that occurs in more than half of the elements of A.

For example, consider array A such that

A[0] = 3 A[1] = 4 A[2] = 3 A[3] = 2 A[4] = 3 A[5] = -1 A[6] = 3 A[7] = 3

The dominator of A is 3 because it occurs in 5 out of 8 elements of A (namely in those with indices 0, 2, 4, 6 and 7) and 5 is more than a half of 8.

Write an efficient algorithm for the following assumptions:

N is an integer within the range [0..100,000];

each element of array A is an integer within the range [1..2,147,483,647].

Note : The file/class should be named as "DominatorElement" with appropriate file extension.

Input Format :

First line of input consists of a single integer 'N' which is the size of the array.

Next N lines consists of N integers which are part of the array A.

Output Format :

Output consists of a single integer which is the dominator of the given array. If no such element is found print -1.

Test case 1:

Input :

8

2

34

33

11

33

44

44

233

Output :

-1

Test case 2:

Input :

10

2

3

2

3

2

3

2

3

3

2

Output :

-1

Test case 3:

Input :

10

2

3

2

2

3

3

2

2

2

7

Output :

2

Test case 4:

Input :

5

13

14

13

11

13

Output :

13

5. Numbers divisible by 'K' in a range.

Write a program that given three integers A, B and K, prints the number of integers within the range [A - B] that are divisible by K.

For example, for A = 6, B = 11 and K = 2, your function should return 3, because there are three numbers divisible by 2 within the range [6..11], namely 6, 8 and 10.

Write an efficient algorithm for the following assumptions:

A and B are integers within the range [0..2,000,000,000];

K is an integer within the range [1..2,000,000,000];

A <= B.

Note : The file/class should be named as "DivisibleByKInRange" with appropriate file extension.

Input Format :

The first line of input consists of a single Integer A (lower range).

The second line of input consists of a single Integer B (upper range).

The third line of input consists of a single Integer K.

Output Format :

Output consists of a single integer which is the number of integers divisible by K within the range A-B.

Test Case 1:

Input :

5

15

6

Output :

2

Test Case 2:

Input :

2

9

1

Output :

8

Test Case 3:

Input :

2

10

2

Output :

5

Test Case 4:

Input :

8

23

4

Output :

4

Test Case 5:

Input :

10

20

8

Output :

1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Tricky Questions**

1. Write a program to Swap two numbers without using temporary variable.

Note : The file/class should be named as "swapWithNoTemp" with appropriate file extension.

Conditions :

X, Y any finite positive value

Input Format :

X - first value

Y - second value

Output Format :

X - value of X after swap

Y - value of Y after swap

Sample IO :

Test Case:

Input :

2

15

Output :

15

2

Test Case2:

Input :

23

45

Output :

45

23

2. Write a program to determine whether a given number X is a part of the Fibonacci Sequence. [0,1,1,2,3,5...]

Note : The file/class should be named as "IsFibonacciNum" with appropriate file extension.

Conditions :

0 <= X <= 1000

Input Format :

Input consists of only one Integer which is X.

Output Format :

Print True if X is present in the Fibonacci series. Print False if X is not present.

Sample IO :

Test Case1:

Input :

8

Output:

True

Test Case2:

Input :

233

Output :

True

Test Case3:

Input :

4123

Output :

False

Test Case4:

Input :

17711

Output :

True

3. Write a C program to find the largest prime factor of a given number 'X'.

Note : The file/class should be named as "LargestPrimeFactor" with appropriate file extension.

Conditions :

0 <= X <= 1000

Input Format :

The first line of input consists of only one Integer which is X.

Output Format :

Output consists of a single line which is the largest prime factor of the given input 'X'.

Test Case1:

Input:

6

Output:

3

Test Case2:

Input:

15

Output:

5

Test Case3:

Input:

127

Output:

127

Test Case4:

Input:

287

Output:

41

4. Write a C program to check whether a number 'X' is palindrome or not.

Note : The file/class should be named as "IsPalindromeNum" with appropriate file extension.

Conditions :

0 <= X <= 9999999

Input Format :

The first line of input consists of only one Integer which is X.

Output Format :

Print True if X is a palindrome number. Print False if X is not a palindrome number.

Test Case1:

Input:

1001

Output:

True

Test Case2:

Input:

1021

Output:

False

Test Case3:

Input:

12321

Output:

True

Test Case4:

Input:

123432

Output:

False

5. Write a C program to convert the given number to its roman numeral.

Note : The file/class should be named as "ToRomanNum" with appropriate file extension.

Conditions :

0 <= X <= 10000

Input Format :

The first line of input consists of only one Integer which is X.

Output Format :

Print the Roman Numeral corresponding to X.

Test Case1:

Input :

9

Output :

IX

Test Case2:

Input :

40

Output :

XL

Test Case3:

Input :

1904

Output :

MCMIV

6. Write a program to multiply the number 'X' by 'Y' without using \* operator.

Note : The file/class should be named as "TrickyMultiply" with appropriate file extension.

Conditions :

-1000 <= X <= 1000

-1000 <= Y <= 1000

Input Format :

The first line of input consists of an Integer which is X.

The second line of input consists of an Integer which is Y.

Output Format :

Print the product of X and Y calculated without using \* sign.

Test Case1:

Input :

4

7

Output:

28

Test Case2:

Input :

10

7

Output:

70

Test Case3:

Input :

24

13

Output:

312

Test Case4:

Input :

-52

77

Output:

-4004

7. Write a C program to Count the number of Trailing Zeros in integer using bitwise operators.

Note : The file/class should be named as "TrailingZeroes" with appropriate file extension.

Conditions :

0 <= X <= 99999

Input Format :

The first line of input consists of an Integer which is X.

Output Format :

Print the number of trailing zeros in the binary format of 'X'.

Test Case1:

Input :

60

Output :

2

Test Case2:

Input :

156

Output :

2

Test Case3:

Input :

22920

Output :

3

Test Case4:

Input :

512

Output :

9

8. Given a positive integer n, the problem is to print if the number has first and last bits as the only set bits.

Note : The file/class should be named as "EnclosedByOne" with appropriate file extension.

Conditions :

0 <= X <= 99999

Input Format :

The first line of input consists of an Integer which is X.

Output Format :

Print "Yes" if the number has first and last bits as the only set bits else print "No".

Test Case1:

Input : 9

Output : Yes

(9)10 = (1001)2, only the first and

last bits are set.

Test Case2:

Input : 15

Output : No

(15)10 = (1111)2, except first and last

there are other bits also which are set.

Test Case3:

Input : 657

Output : No

Test Case4:

Input : 513

Output : Yes

9. Write a C program to divide a given number 'X' (Dividend) by 'Y' (Divisor) without using '/' and '–' operators

Note : The file/class should be named as "TrickyDivide" with appropriate file extension.

Conditions :

-1000 <= X <= 1000

-1000 <= Y <= 1000

X >= Y

Input Format :

The first line of input consists of an Integer which is X (Dividend).

The second line of input consists of an Integer which is Y (Divisor).

Output Format :

Print the Quotient of X divided by Y calculated without using '/', '-' signs.

Test Case1:

Input :

-11

3

Output :

-3

Test Case2:

Input :

10

10

Output :

1

Test Case3:

Input :

-6

-2

Output :

3

Test Case4:

Input :

325

15

Output :

21

10. Write a program given a positive fractional number n, round-off this number to a given no. of significant digits, d.

Note : The file/class should be named as "RoundOffFloat" with appropriate file extension.

Conditions :

0.000000 < n <= 1000.000000

d > 0

Input Format :

The first line of input consists of an fractional number which is n.

The second line of input is of an Integer which is d.

Output Format :

Print the number n rounded off by d digits.

Test Case1:

Input :

139.5912

4

Output :

139.6

Test Case2:

Input :

24.07182

5

Output :

24.072

Test Case3:

Input :

33.57

2

Output :

34

11. Write a c program to find if a given number 'X' is even or odd without using % operator

Note : The file/class should be named as "EvenOrOdd" with appropriate file extension.

Conditions :

0 <= X <= 99999

Input Format :

The first line of input consists of an Integer 'X'.

Output Format :

Print "Even" if 'X' is an even number. Print "Odd" if 'X' is an odd number.

Test Case1:

Input:

121

Output:

Odd

Test Case2:

Input:

232

Output:

Even

Test Case3:

Input:

23485

Output:

Odd

Test Case4:

Input:

9132

Output:

Even

12. Given a number n, print the following pattern without using any loop.

Note : The file/class should be named as "NumSeries01" with appropriate file extension.

Conditions :

1<= n <=1000

Input Format :

Single integer value 'n'

Output Format :

number series with the following pattern

Test Case1:

Input :

16

Output :

16 11 6 1 -4 1 6 11 16

Test Case2:

Input :

10

Output :

10 5 0 5 10

Test Case3:

Input :

31

Output :

31 26 21 16 11 6 1 -4 1 6 11 16 21 26 31

13. Write a program to check if two numbers are equal without using arithmetic operators or comparison operators.

Note : The file/class should be named as "CheckIfEqual" with appropriate file extension.

Conditions :

0 <= X <= 99999

0 <= Y <= 99999

Input Format :

The first line of input consists of an Integer 'X'.

The second line of input consists of an Integer 'Y'.

Output Format :

Print "Equal" if X=Y else print "Not equal".

Test Case1:

10

10

Equal

Test Case2:

10

20

Not equal

Test Case3:

17

17

Equal

Test Case4:

17

27

Not equal

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Daily Coding Problems**

1. Word Puzzle

This problem was asked by Coursera.

Given a 2D board of characters and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

For example, given the following board:

[

['A','B','C','I'],

['S','I','C','S'],

['E','D','E','E']

]

Input Format :

Input consists of a single string in UPPER CASE.

Output Format :

Output consists of a single line stating "True" - if the board contains the input string or "False" - if it doesn't contain the input string.

Note : The file/class should be named as "ArrayWordPuzzle" with appropriate file extension.

Input 1 :

ABCCED

Output 1 :

True

Input 2 :

ABCB

Output 2 :

False

Input 3 :

SEE

Output 3 :

True

Input 4 :

SICK

Output 4 :

False

2. Longest consecutive number sequence

This problem was asked by Microsoft.

Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

For example, given [100, 4, 99, 1, 3, 2, 200], the longest consecutive element sequence is [1, 2, 3, 4].

Print its length: 4.

Your algorithm should run in O(n) complexity.

Input Format :

First line of the input is an integer N which is the size of the array.

Next N lines of input contains a single integer each representing elements of the array arr[]

Output Format :

Single integer which is the length of the longest consecutive elements sequence.

Constraints :

2 <= N <= 100

0 <= arr[i] <= 10000

Note : The file/class should be named as "LongestConsecutiveSequence" with appropriate file extension.

Input 1 :

7

100

4

99

1

3

2

200

Output 1 :

4

Input 2 :

11

36

41

56

35

44

33

34

92

43

32

42

Output 2 :

5

3. Pogo Jump 1

This problem was asked by Google.

You are in an infinite 2D grid where you can move in any of the 8 directions:

(x,y) to

(x+1, y),

(x - 1, y),

(x, y+1),

(x, y-1),

(x-1, y-1),

(x+1,y+1),

(x-1,y+1),

(x+1,y-1)

You are given a sequence of points and the order in which you need to cover the points. Give the minimum number of steps in which you can achieve it. You start from the first point.

Example:

Input: [(0, 0), (1, 1), (1, 2)]

Output: 2

Move from (0, 0) to (1, 1) in 1 step(diagonal) and then from (1, 1) to (1, 2) in 1 step (rightwards)

Input Format :

First line of the input is an integer N which is the number of points.

Next N lines of input contains 2 integers each separated by a space representing x and y of each point.

Output Format :

Single integer which is the length of the longest consecutive elements sequence.

Constraints :

1 <= N <= 100

0 <= x,y

Note : The file/class should be named as "PogoJumpMinimum" with appropriate file extension.

Input 1 :

3

0 0

1 1

1 2

Output 1 :

2

Input 2 :

4

4 6

1 2

4 5

10 12

Output 2 :

14

4. Pogo Jump 2

This problem was asked by Pinterest.

Given an integer list where each number represents the number of hops you can make, determine whether you can reach to the last index starting at index 0.

For example, [2, 0, 1, 0] returns True while [1, 1, 0, 1] returns False.

Input Format :

First line of the input is an integer N which is the number of points.

Next N lines of input each contains an integer representing the possible distance of jump from that point.

Output Format :

Single line - "True" if able to reach the end of the array / "False" if not able to reach the end of the array.

Constraints :

1 <= N <= 100

0 <= arr[i] <= 10000

Note : The file/class should be named as "PogoJumpToEnd" with appropriate file extension.

Input 1 :

4

2

0

1

0

Output 1 :

True

Input 2 :

4

1

1

0

1

Output 2 :

False

Input 3 :

4

0

1

1

0

Output 3 :

False

5. Write a program to find if a string is the rotated version of another string.

This problem was asked by Google.

Given two strings A and B, return whether or not A can be shifted some number of times to get B.

For example, if A is abcde and B is cdeab, print true.

Input Format :

First line of the input is string A.

Second line of the input is another string B.

Output Format :

Single line - "True" if it is possible to get string B upon rotating string A by 'n' times otherwise "False".

Constraints :

1 <= |S| <= 100

Note : The file/class should be named as "IsStringRotated" with appropriate file extension.

Input 1 :

abcde

cdeab

Output 1 :

True

Input 2 :

abc

acb

Output 2:

False

Input 3 :

amazon

azonam

Output :

True

Input 4 :

amazon

omanaz

Output :

False

6. Write a program to swap even and odd bits of a number represented in binary.

This problem was asked by Cisco.

Given an unsigned integer, swap its even and odd bits. The 1st and 2nd bit should be swapped, the 3rd and 4th bit should be swapped, and so on.

For example, 10101010 should be 01010101. 11100010 should be 11010001.

Input Format :

Input consists of a single integer N.

Output Format :

Single integer of bit-swapped input.

Constraints :

1 <= N <= 10000

Note : The file/class should be named as "SwapOddEvenBits" with appropriate file extension.

Input 1 :

170

Output 1 :

85

Input 2 :

226

Output 2 :

209

Input 3 :

345

Output 3 :

678

7. Reverse words in a sentence.

This problem was asked by Google.

Given a string of words delimited by spaces, reverse the words in string. For example, given "hello world here", return "here world hello"

Taking a mutable string representation, perform this operation in-place.

Input Format :

Input contains a single line which is the input sentence A.

Output Format :

Input contains a single line which is the reverse form of input sentence A.

Constraints :

1 <= |S| <= 1000

Note : The file/class should be named as "ReverseSentence" with appropriate file extension.

Input 1 :

hello world

Output 1 :

world hello

Input 2 :

welcome to sri shakthi

Output 2 :

shakthi sri to welcome

Input 3 :

string of words delimited by

Output 3 :

by delimited words of string

8. Square array elements and sort the resulting array.

This problem was asked by Google.

Given a sorted list of integers, square the elements and give the output in sorted order.

For example, given [-9, -2, 0, 2, 3], return [0, 4, 4, 9, 81].

Input Format :

First line of the input is an integer N which is the number of elements in the input array.

Next N lines of input each contains an integer representing the elements of the array.

Output Format :

Single line of output consisting of N space separated integers.

Constraints :

1 <= N <= 100

0 <= arr[i] <= 10000

Note : The file/class should be named as "SquareAndSortArray" with appropriate file extension.

Input 1 :

5

-9

-2

0

2

3

Output 1 :

0 4 4 9 81

Input 2 :

6

-6

-3

-1

2

4

5

Output 2 :

1 4 9 16 25 36

Input 3 :

5

-5

-4

-2

0

1

Output 3 :

0 1 4 16 25

9. Calculate minimum stab spanning all line segments.

This problem was asked by Google.

Given a set of closed intervals, find the smallest set of numbers that covers all the intervals. If there are multiple smallest sets, return any of them.

For example, given the intervals [0, 3], [2, 6], [3, 4], [6, 9], one set of numbers that covers all these intervals is {3, 6}.

Input Format :

First line of the input is an integer N which is the number of elements in the input array.

Next N lines of input each contains 2 space-separated integers representing the start and end points of a line segment.

Output Format :

Single line of output consisting of N space separated integers.

Constraints :

1 <= N <= 100

0 <= start, end <= 10000

Note : The file/class should be named as "MinimumStabCalc" with appropriate file extension.

Input 1 :

4

0 3

2 6

3 4

6 9

Output 1 :

3 6

Input 2 :

5

4 8

11 13

1 4

6 12

10 15

Output 2 :

4 11

Input 3 :

6

0 2

2 0

1 2

1 0

0 0

2 1

Output 3 :

0 1

10. Minimum coin denominations

This problem was asked by Google.

Find the minimum number of coins required to make n cents.

You can use standard American denominations, that is, 1¢, 5¢, 10¢, and 25¢.

For example, given n = 16, return 3 since we can make it with a 10¢, a 5¢, and a 1¢.

Input Format :

First line of the input is an integer n.

Output Format :

Single integer which is the minimum number of coins.

Constraints :

1 <= N <= 1000

Note : The file/class should be named as "MinimumCoins" with appropriate file extension.

Input 1 :

16

Output 1 :

3

Input 2 :

90

Output 2 :

5

Input 3 :

100

Output 3 :

4

11. Partition the given array based on a pivot value.

This problem was asked by Amazon.

Given a pivot x, and a list lst, partition the list into three parts.

The first part contains all elements in lst that are less than x

The second part contains all elements in lst that are equal to x

The third part contains all elements in lst that are larger than x Ordering within a part can be arbitrary.

For example, given x = 10 and lst = [9, 12, 3, 5, 14, 10, 10], one partition may be [9, 5, 3, 10, 10, 14, 12].

Input Format :

First line of the input is an integer N which is the number of elements in the input array arr[].

Next N lines of input each contains an integer representing the elements arr[i].

Last line of input is a single integer which corresponds to the pivot value.

Output Format :

Single line consisting of N space separated integers.

Constraints :

1 <= N <= 100

0 <= arr[i] <= 10000

Note : The file/class should be named as "ArrayPartition" with appropriate file extension.

Input 1 :

7

9

12

3

5

14

10

10

10

Output 1 :

9 5 3 10 10 14 12

Input 2 :

7

9 12 3 5 14 10 10

8

Output 2 :

5 3 14 12 10 10 9

Input 3 :

3

8

8

8

8

Output 3 :

8 8 8

Input 4 :

2

3

5

8

Output 4 :

5 3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Problem Solving Problems**

1. Find the Missing Number

You are given a list of n-1 integers and these integers are in the range of 1 to n. There are no duplicates in list. One of the integers is missing in the list. Write an efficient code to find the missing integer. If no number is missing print -1.

File / Class name should be "missingNumber".

Input Format :

N - size of the array

'N' integer values

Output Format :

Single integer which is missing in the array

Sample IO :

Input :

7

1 2 4 6 3 7 8

Output :

5

2. There is an integer array with duplicate elements. Print the unique element which is not repeated. If more than one elements are found unique print the one with the least value.

File / Class name should be "nonRepeatNum".

Input Format :

N - size of the array

'N' integer values

Output Format :

Single integer from the array which is the unique number

Sample IO :

Input :

9

1 1 2 2 3 4 4 5 5

Output :

3

3. How to find kth smallest element in an unsorted array?

You are given an unsorted array of numbers and k, you need to find the kth smallest number in the array.

One way to solve this problem is to sort the array in ascending order then pick the k-1th element, that would be your

kth smallest number in array because array index starts at zero, but can you do better?

File / Class name should be "kthSmallest".

Input Format :

N - size of the array

'N' integer values

k - smallest number corresponding to the index in ascending order.

Output Format :

Single integer from the array which is the unique number

Sample IO :

Input :

6

10 2 5 3 9 4

2

Output :

3

4. How to find common elements in three sorted array?

Given three arrays sorted in non-decreasing order, print all common elements in these arrays.

File / Class name should be "commonElements".

Input Format :

N1 - size of the array1

'N1' integer values

N2 - size of the array2

'N2' integer values

N3 - size of the array3

'N3' integer values

Output Format :

Set of integers common to the above arrays.

Input :

6

1 5 10 20 40 80

5

6 7 20 80 100

8

3 4 15 20 30 70 80 120

Output :

20 80

5. How to find the first repeating element in an array of integers?

Given an array of integers, find the first repeating element in it. We need to find the element that occurs more than once and whose index of the first occurrence is smallest.

File / Class name should be "firstRepeating".

Input Format :

N - size of the array

'N' integer values

Output Format :

Single integer from the array which is the first-repeating

Input:

7

10 5 3 4 3 5 6

Output:

5

6. How to find if there is a sub array with sum equal to zero?

Here you are given an array of positive and negative numbers, find if there is a sub-array of continuous elements with 0 sum.

File / Class name should be "subArrayZero".

Input Format :

N - size of the array

'N' integer values

Output Format :

true - if sum of any sub-array of continuous elements is zero

false - otherwise

Input :

5

4 2 -3 1 6

Output:

true

7. How to reverse an array in place?

You need to write a program which accepts an integer array and your program needs to reverse that array in place, which means you cannot use additional buffer or array, but one or two variables will be fine.

Of course you cannot use any open source library or method to directly solve this problem and you should not print the array in reverse directly. You need to create your own logic

File / Class name should be "arrayRevInPlace".

Input Format :

N - size of the array

'N' integer values

Output Format :

Input array in reverse order

Input :

7

10 5 3 4 3 5 6

Output :

6

5

3

4

3

5

10

8. Rearrange an array such that arr[i] = i

Given an array of elements of length N, ranging from 1 to N. All elements may not be present in the array. If element is not present then there will be -1 present in the array. Rearrange the array such that A[i] = i and if i is not present, display -1 at that place.

File / Class name should be "valueInIndex".

Input Format :

N - size of the array

'N' integer values

Output Format :

Input array in proper arrangement

Input :

6

6 1 9 3 2 4

Output :

-1

1

2

3

4

-1

6

-1

-1

9

9. Segregate 0s and 1s in an array

You are given an array of 0s and 1s in random order. Segregate 0s on left side and 1s on right side of the array. Traverse array only once.

File / Class name should be "arrangeOneZero".

Input Format :

N - size of the array

'N' integers(0 or 1)

Output Format :

Input array in left-right arrangement

Input :

10

0 1 0 1 0 0 1 1 1 0

Output :

0 0 0 0 0 1 1 1 1 1