Code -collections

1. Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums [0] + nums [1] == 9, we return [0, 1].

Example 2:

Input: nums = [3,2,4], target = 6

Output: [1,2]

Example 3:

Input: nums = [3,3], target = 6

Output: [0,1]

Solution:

```
class Main {
    public static void main(String[] args) {
        int[] arr = {2, 7, 11, 15};
        int target = 9;

        for (int i = 0; i < arr[i]; i++) {
            for (int j = i + 1; j < arr[j]; j++)
             {
                if (arr[i] + arr[j] == target) {
                    System.out.println("\n hence the Array will be : [" + arr[i] + ", " +
arr[j] + "]");
                    System.out.println("AT Index: [" + i + ", " + j + "]");
                    return;
                }
            }
        }

        System.out.println("No solution found.");
    }
}
```

2. Chef wants to buy all the N items in a shop. The price of the i-th item is given by Ai . There is also a discount coupon that costs X rupees and reduces the cost of every item by Y rupees. If the price of an item is less than or equal to Y, it becomes free (costs 0).

Chef will buy the discount coupon if and only if the total price he pays after buying the coupon is strictly less than the price he pays without buying the coupon.

Test case 1:

1: The original cost of the items is 15+8+22+6=51. Buying the coupon costs 30, and after buying it the cost of buying all the items is

5+0+12+0=17. The total cost of buying everything with the coupon is 30+17=47, which is strictly less than 51. So, Chef will buy the coupon.

Test case 2: The original cost of the items is 15+8+22+6=51. Buying the coupon costs

40, and after buying it the cost of buying all the items is 5+0+12+0=17. The total cost of buying everything with the coupon is 40+17=57, which is more than 51. So, Chef will not buy the coupon.

[01-01-2025 22:54] +91 94211 75379: #include <stdio.h>

```
int main() {
   int t;
   scanf("%d", &t);

   while (t--) {
      int n, x, y;
      scanf("%d %d %d", &n, &x, &y);
      int a[n];

      for (int i = 0; i < n; i++) {
         scanf("%d", &a[i]);
      }
```

```
        // Your code goes here


        int sum1 = 0; // for original sum
        int sum2 = 0; // for discounted sum


        for(int i = 0; i<n; i++){
            sum1 += a[i];
            if(a[i] - y < 0){
                sum2 += 0;
            }else{
                sum2 += a[i] - y;
            }
        }
        if((sum2 + x) < sum1){
            printf("COUPON\n");
        }
        else{
            printf("NO COUPON\n");
        }
    }
    return 0;
}
```

3. There is a string that consists of lowercase english alphabets and we have to make the string a palindrome

For this, we have 2 types of operations

1. Re-arrange the string however you want (this operation is free)

2. Add a new character to the end of the string (this operation will cost 1Re)

Your task is to calculate the minimum amount of money to make the string a palindrome

[01-01-2025 23:02] +91 93078 02090: Test Case 1 (abcde):

Frequencies: a: 1, b: 1, c: 1, d: 1, e: 1 (all characters have odd frequencies).

We have 5 odd frequencies, so we need to add 4 characters to make the string a palindrome.

Result: 4.

Test Case 2 (aabb):

Frequencies: a: 2, b: 2 (all characters have even frequencies).

No characters need to be added because the string can already be rearranged into a palindrome.

Result: 0

[01-01-2025 23:03] +91 93078 02090:

```
 #include <iostream>
#include <string>
using namespace std;
int minCostToMakePalindrome (int N, const string& str) {
   for (char c : str) {
      freq[c]++;
```

```
    }
    int oddCount = 0;
    for (auto& entry : freq) {
        if (entry.second % 2 != 0) {
            oddCount++;
        }
    }
    return max(0, oddCount - 1);
}

int main() {
    int T;
    cin >> T;

    while (T--) {
        int N;
        cin >> N;
        string str;
        cin >> str;
        cout << minCostToMakePalindrome (N, str) << endl;
    }
    return 0;
}
```

Given an integer x, return true if x is a

palindrome

, and false otherwise.

Example 1:

Input: x = 121

Output: true

Explanation: 121 reads as 121 from left to right and from right to left.

Example 2:

Input: x = -121

Output: false

Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

Example 3:

Input: x = 10

Output: false

Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

https://leetcode.com/problems/palindrome-number.

Solution:

```
class Solution {
    public boolean isPalindrome(int x) {
        if (x < 0) {
            return false;
        }
        int temp = x, rev = 0;
        while (x > 0) {
            int rem = x % 10;
            rev = rev * 10 + rem;
            x = x / 10;
        }
        return temp == rev;
    }
}
```

[03-01-2025 21:09] +91 93078 10328: Given an array arr[] where no two adjacent elements are same, find the index of a peak element. An element is considered to be a peak if it is greater than its adjacent elements (if they exist). If there are multiple peak elements, return index of any one of them. The output will be "true" if the index returned by your function is correct; otherwise, it will be "false".

Note: Consider the element before the first element and the element after the last element to be negative infinity.

Examples :

Input: arr = [1, 2, 4, 5, 7, 8, 3]

Output: true

Explanation: arr[5] = 8 is a peak element because arr[4] < arr[5] > arr[6].

[03-01-2025 21:10] +91 93078 10328: class Solution {

```
public int peakElement(int[] arr) {

  int n = arr.length;

  if(n == 1)
  return 0;

  for(int i=0; i<n; i++)
  {
    if(i == 0)
    {
      if(arr[i] > arr[i+1])
      return i;
    }


    else if(i == n-1)
    {
      if(arr[i] > arr[i-1])
      return i;
    }


    else
    {
      if(arr[i] > arr[i-1] && arr[i] > arr[i+1])
      return i;
    }
```

```
        }

        return -1;
    }
}
```

[01-01-2025 22:51] +91 94211 75379: Chef wants to buy all the N items in a shop. The price of the i-th item is given by Ai . There is also a discount coupon that costs X rupees and reduces the cost of every item by Y rupees. If the price of an item is less than or equal to

Y, it becomes free (costs 0).

Chef will buy the discount coupon if and only if the total price he pays after buying the coupon is strictly less than the price he pays without buying the coupon.

[01-01-2025 22:54] +91 94211 75379: Test case 1

1: The original cost of the items is 15+8+22+6=51. Buying the coupon costs 30, and after buying it the cost of buying all the items is

5+0+12+0=17. The total cost of buying everything with the coupon is 30+17=47, which is strictly less than 51. So, Chef will buy the coupon.

Test case 2: The original cost of the items is 15+8+22+6=51. Buying the coupon costs

40, and after buying it the cost of buying all the items is 5+0+12+0=17. The total cost of buying everything with the coupon is 40+17=57, which is more than 51. So, Chef will not buy the coupon.

[01-01-2025 22:54] +91 94211 75379: #include <stdio.h>

```
int main() {
    int t;
    scanf("%d", &t);

    while (t--) {
        int n, x, y;
        scanf("%d %d %d", &n, &x, &y);
        int a[n];

        for (int i = 0; i < n; i++) {
            scanf("%d", &a[i]);
        }
        // Your code goes here

        int sum1 = 0; // for original sum
        int sum2 = 0; // for discounted sum

        for(int i = 0; i<n; i++){
            sum1 += a[i];
            if(a[i] - y < 0){
                sum2 += 0;
            }else{
                sum2 += a[i] - y;
            }
        }
        if((sum2 + x) < sum1){
            printf("COUPON\n");
        }
        else{
            printf("NO COUPON\n");
        }
    }
    return 0;
}
```

[01-01-2025 23:01] +91 93078 02090: Problem

There is a string that consists of lowercase english alphabets and we have to make the string a palindrome

For this, we have 2 types of operations

1. Re-arrange the string however you want (this operation is free)

2. Add a new character to the end of the string (this operation will cost 1Re)

Your task is to calculate the minimum amount of money to make the string a palindrome

[01-01-2025 23:02] +91 93078 02090: Test Case 1 (abcde):

Frequencies: a: 1, b: 1, c: 1, d: 1, e: 1 (all characters have odd frequencies).

We have 5 odd frequencies, so we need to add 4 characters to make the string a palindrome.

Result: 4.

Test Case 2 (aabb):

Frequencies: a: 2, b: 2 (all characters have even frequencies).

No characters need to be added because the string can already be rearranged into a palindrome.

Result: 0

```
[01-01-2025 23:03] +91 93078 02090: #include <iostream>
#include <string>
using namespace std;
int minCostToMakePalindrome(int N, const string& str) {
    for (char c : str) {
        freq[c]++;
    }
    int oddCount = 0;
    for (auto& entry : freq) {
        if (entry.second % 2 != 0) {
            oddCount++;
        }
    }
    return max(0, oddCount - 1);
}

int main() {
    int T;
    cin >> T;

    while (T--) {
        int N;
        cin >> N;
        string str;
        cin >> str;
        cout << minCostToMakePalindrome(N, str) << endl;
    }
    return 0;
}
```

[01-01-2025 23:33] +91 96992 28162: Input Format

The first line contains a single integer, , denoting the size of the array.

Each line of the subsequent lines contains a single integer denoting the value of element .

Output Format

You are not responsible for printing any output to stdout. Locked code in the editor loops through array and prints each sequential element on a new line.

Sample Input

5

```
10

20

30

40

50
```

Sample Output

```
10

20

30

40

50
```

[01-01-2025 23:34] +91 96992 28162: import java.util.*;

```
public class Solution {

    public static void main(String[] args) {

        Scanner scan = new Scanner(System.in);
        int n = scan.nextInt();
        int a[]=new int [n];
        for(int i=0;i<n;i++){
            a[i]=scan.nextInt();
        }
        scan.close();

        // Prints each sequential element in array a
        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
}
```

[02-01-2025 16:14] +91 94211 75379: Problem

You are required to enter a word that consists of x and y that denote the number of Zs and Os respectively. The input word is considered similar to word zoo if 2 * x = y.

Determine if the entered word is similar to word zoo.

For example, words such as zzoooo and zzzoooooo are similar to word zoo but not the words such as zzooo and zzzooooo.

[02-01-2025 16:14] +91 94211 75379: #include <stdio.h>

```
int main(){
	char ch[20];
	int c1 = 0, c2;
	gets(ch);
	int i = 0;
	while(ch[i] != '\0') i++;
	c2 = i;
	i=0;
	while(ch[i] != 'o'){
		c1++;
		i++;
	}
	c2 = c2 - c1;

	if(2*c1 == c2){
		printf("Yes");
	}
	else{
```

```
                printf("No");
        }
}
```

[02-01-2025 20:42] +91 93078 02090: Title: For encoding an even-length binary string into a sequence of A, T, C, and G, we iterate from left to right and replace the characters as follows:

00 is replaced with A

01 is replaced with T

10 is replaced with C

11 is replaced with G

Given a binary string S of length N (N is even),find the encoded sequence.
Link:https://www.codechef.com/practice/course/strings/STRINGS/problems/DNASTORAGE
Solution:

```
#include <bits/stdc++.h>
#include <string>
using namespace std;
string encodedBinaryString(const string &S){
   string encoded="";
   for(size_t i=0;i<S.length();i+=2){
       string store=S.substr(i,2);
         if(store == "00"){
            encoded += "A";
         }
        else if(store == "01"){
            encoded += "T";
         }
        else  if(store == "10"){
            encoded += "C";
         }
         if(store == "11"){
            encoded += "G";
```

```
          }


        }

        return encoded;

    }




int main() {

    int t;

    cin>>t;

    while(t--){

        int n;

        cin>>n;

        string S;

        cin>>S;

        cout<<encodedBinaryString(S)<<endl;


        // your code goes here

    }

    return 0;


}
```

Test Cases :
For Input:

```
4

2

00

4

0011
```

```
6

101010

4

1001
Your Output :
A

AG

CCC

CT
```

[02-01-2025 22:15] +91 93078 10328: Write a function that takes an integer minutes and converts it to seconds.

[02-01-2025 22:15] +91 93078 10328: class Challenge {

```
                    public static int convert(int minutes) {

                    int second =0;

                    second=minutes*60;

                    return second;

  }


}

class conversion

{

          public static void main(String []args)

          {

                    int x=5;

                    Challenge ob=new Challenge();

                    System.out.println(ob.convert(x));

          }

}
```

[02-01-2025 23:52] +91 96992 28162: Task

Given an integer,n, perform the following conditional actions:

If n is odd, print Weird

If n is even and in the inclusive range of 2to5 , print Not Weird

If n is even and in the inclusive range of to , print Weird

If n is even and greater than , 20 print Not Weird

Complete the stub code provided in your editor to print whether or not is weird.

Input Format

A single line containing a positive integer, .

Constraints

Output Format

Print Weird if the number is weird; otherwise, print Not Weird.

Sample Input 0

3

Sample Output 0

Weird

Sample Input 1

24

Sample Output 1

Not Weird

Sample Output 0

```
Weird
```

Sample Input 1

```
24
```

Sample Output 1

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.regex.*;

public class Solution {



    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        int N = scanner.nextInt();
        if(N%2!=0)
        {
            System.out.println("Weird");
        }
        else{
//if(N%2!=0)
        {
            if(N>=2 && N<=5)
            {
                System.out.println("Not Weird");
            }
            else if(N>20)
            {
                 System.out.println("Not Weird");
            }


        }






        scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");


        scanner.close();
    }
}
}
```

You are given a string s consisting of lowercase English letters, and an integer k. Your task is to convert the string into an integer by a special process, and then transform it by summing its digits repeatedly k times. More specifically, perform the following steps:

1. **Convert s into an integer by replacing each letter with its position in the alphabet (i.e. replace 'a' with 1, 'b' with 2, ..., 'z' with 26).**
2. **Transform the integer by replacing it with the sum of its digits.**
3. **Repeat the transform operation (step 2) k times in total.**

For example, if s = "zbax" and k = 2, then the resulting integer would be 8 by the following operations:

1. **Convert: "zbax" ➝ "(26)(2)(1)(24)" ➝ "262124" ➝ 262124**
2. **Transform #1: 262124 ➝ 2 + 6 + 2 + 1 + 2 + 4 ➝ 17**
3. **Transform #2: 17 ➝ 1 + 7 ➝ 8**

Return the resulting integer after performing the operations described above.

Example :

Input: s = "zbax", k = 2

Output: 8

```
class Main {
   public static void main(String[] args) {
      String[] str = {"zbax"};
      int k = 2;



      int result = trString(str[0], k);
      System.out.println("The sumof (53=5+3) is: "+result);
   }
   public static int trString(String str, int k) {
      int n = 0;
```

```
        for (int i = 0; i < str.length(); i++) {
            n += (str.charAt(i) - 'a' + 1);
        }

        for (int i = 0; i < k; i++) {
            n = digitSum(n);
        }

        return n;
    }
public static int digitSum(int n) {
        int sum = 0;


        while (n > 0) {
            sum += n % 10;
             n /= 10;
        }

        return sum;
    }
}
```

https://leetcode.com/problems/integer-to-roman

Solution:

```
public class Solution {
    public String intToRoman(int num) {
        int[] val = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};
        String[] symbols = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV",
"I"};
        StringBuilder roman = new StringBuilder();
        for (int i = 0; i < val.length; i++) {

            while (num >= val[i]) {
                roman.append(symbols[i]);
                num -= val[i];
            }
        }
        return roman.toString();
    }
}
```

Note: had referred gpt in order to take help to append the string without creating the new function for it .

https://leetcode.com/problems/longest-common-prefix

Solution:

```
class Solution {
    public String longestCommonPrefix(String[] strs) {
        if (strs == null || strs.length == 0) {
            return "";
        }
        String prefix = strs[0];
        for (int i = 1; i < strs.length; i++) {
            int j = 0;
            while (j < prefix.length() && j < strs[i].length() && prefix.charAt(j) ==
strs[i].charAt(j)) {
                j++;
            }
            prefix = prefix.substring(0, j);
            if (prefix.isEmpty()) {
                return "";
            }
        }

        return prefix;
    }
}
```

https://leetcode.com/problems/search-insert-position

Solution:

```
class Solution {
    public int searchInsert(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return left;
    }
}
```

```
class Main {
    public static void main(String[] args) {
        Solution ob = new Solution();


        int[] nums = {1, 3, 5, 6};
        int target1 = 5;



        System.out.println("Index for target " + target1 + ": " +
ob.searchInsert(nums, target1));


    }
}
```