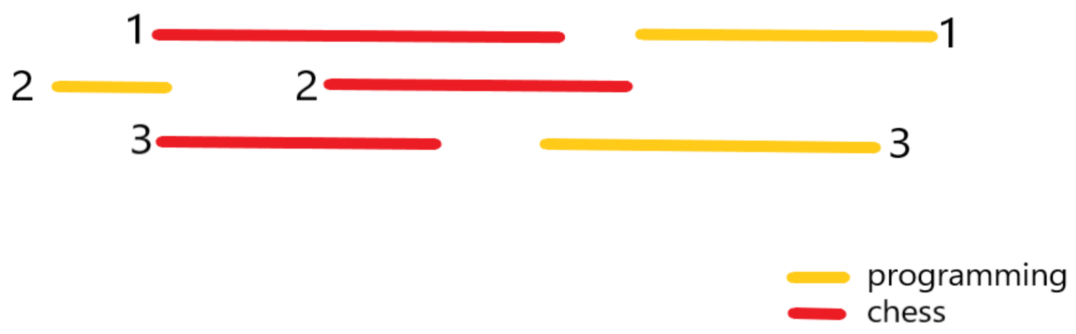
Problem A

We need to maximize the distance between chess class and programming class.



if we take (3rd one in chess and 1st in programming) we will get the maximum distance.

So, we need to get four values (first end of chess, first end of programming, last start of chess, last start of programming)

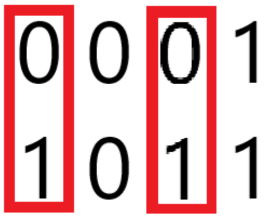
$$Max(\,(last\ start\ programming - first\ end\ chess)\,,$$
$$(last\ start\ chess - first\ end\ programming)\ \ ,0)$$

```
#include <iostream>
using namespace std;
Int main (){
int n ; cin>>n ;
int small_end_chess =INF , great_start_chess = 0 ;
for (int i = 0; i < n; ++i) {
   int x , y ;
    cin>>x>>y;
    small_end_chess = min(small_end_chess , y);
    great_start_chess = max(great_start_chess , x);
}
int m ; cin>>m ;
int small_end_pro =INF , great_start_pro = 0 ;
for (int i = 0; i < m; ++i) {
    int x , y ;
    cin>>x>>y;
    small_end_pro = min(small_end_pro , y);
    great_start_pro = max(great_start_pro , x);
}
cout<< max({(great_start_pro - small_end_chess) , (great_start_chess - small_end_pro)
,0})<<endl;
}
```

Problem B

$D(S\,,T)$ is distance is the number of positions i, such that s_i isn't equal to t_i.

We need to create new string **M** that , $D(M\,,S\,) \;=\; D(M\,,T\,)$



Idea: let's iterate over two strings S, T, and check if bits in same index not equal.

We have two choices

- Put in **M** the S's bit
- Put in **M** the T's bit

But it's **impossible** to get solution $D(S\,,T)$ is odd number, because in first time we pick from S and in second we pick from T and in third we pick from S and in fourth we pick from T.

So, if $D(S\,,T)$ odd it's mean distance in $D(M\,,S\,) \;>\; D(M\,,T\,)$.

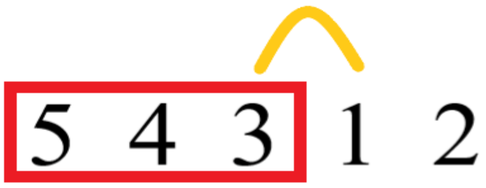
```
#include <iostream>
using namespace std;
Int main (){
    string s1 , s2 ;
    cin>>s1>>s2 ;
    int cnt =  0 ;
    string ans = s1 ;
    for (int i = 0; i < s1.size(); ++i) {
        if(s1[i]!=s2[i])
        {
            if(cnt%2==0){ans[i] = s1[i];}
            else{ans[i]=s2[i];}
            cnt++;
        }
    }
    if(cnt%2==1){cout<<"impossible"; return 0;}
    cout<<ans<<endl;
}
```

Problem C

State	Input	output
No one	1 2 3 4 5	0
5th	5 1 2 3 4	1
4th	4 1 2 3 5	1
3rd 4th 5th	5 4 3 1 2	3

From this table we noticed that if $a_i \geq a_j$ when $J > I$ the index I is new message and all prefix of I.

That's mean if we get value in position I grater than value in position J, all messages before I is new messages [0 , I]



```
#include <iostream>
using namespace std;
Int main (){
    int n ;
    cin>>n ;
    int arr[n] ;
    in(arr,n)
    for (int i = n-1; i >= 1 ; --i) {
        if(arr[i]<arr[i-1])
        {
            return cout<<i , 0 ;
        }
    }
    return cout<<0 , 0 ;
}
```

Problem D

In this problem we need at least in each row 2 cells with value 1, so we have these combinations.

0 1 1
1 0 1
1 1 0
1 1 1

So, we noticed that if sum of all values $Sum(X, Y, Z) > 1$, we can take this problem.

```
#include<iostream>
using namespace std;
int main ()
{
    int a,b,c,n,count=0;
    cin>>n;
    for(int i=0 ; i<n ;i++)
        {
            cin>>a>>b>>c;
            if((a+b+c)> 1)
                count++;
        }
    cout <<count;
    return 0;
}
```

Problem E

In this problem we need to remove any substring contains more than or equal 3 X’s.

So, we will count number of continuous X’s and remove $(n - 2)$

X X X X X X X O X X O X X X

In this example we have 3 substrings

Substring	Number of X’s	How many will be removed
X X X X X X X	7	$Max(7 - 2, 0)$
X X	2	$Max(2 - 2, 0)$
X X X	3	$Max(3 - 2, 0)$

```
#include<iostream>
using namespace std;
int main ()
{
int n ;
    cin>>n ;
    string s  ;
    cin>>s;
    int cnt_x = 0 , ans = 0 ;
    for (int i = 0; i < s.size(); ++i) {
        if(s[i]=='x'){
            while (s[i]=='x')
            {
                cnt_x++;
                i++;
            }
            ans += max(0,cnt_x-2);
            cnt_x = 0 ;
            i--;
        }
    }
    cout<<ans<<endl;}
```

Problem F

In this problem we need to repeat the array to make new array's longest increasing subsequence, for example:

3 2 1 3 2 1 3 2 1

If array is [3, 2, 1] so I will repeat array three times to get longest increasing subsequence.

So, I need the distinct values only (1,2,3) if I have (1, 1, 2, 2, 2, 3) to build my array increasingly will be (1, 2, 3).

```
#include<iostream>
using namespace std;
int main ()
{
int size,count=0,n;
    cin>>n;
    while(n--) {
        cin >> size;
        int arr[size];
        for (int i = 0; i < size; ++i) {
            cin >> arr[i];
        }
        sort(arr, arr + size);
        for (int i = 0; i < size; ++i) {
            if (arr[i] != arr[i + 1]) {
                count++;
            }
        }
        cout<<count<<endl;
        count=0;
    }
}
```

Also, u can use set, frequency array, map.

Problem G

As you know when you divide some numbers by another number the values decreased, for example:

$$\frac{100}{2} < 100$$

If I divide array into two subarrays, it decreases the sum value on two subarrays.

So, if I take in first subarray **Max** value, and in second subarray sum- Max, I will get the maximum.

$$\boxed{\frac{MAX}{1}} + \boxed{\frac{sum - MAX}{n - 1}}$$

```
#include<iostream>
using namespace std;
int main ()
{
int tc; cin>>tc;
    while (tc--)
    {
        long long n , mexo=-1e9+1;
        long long sum=0;
        cin>>n;
        for (int i = 0; i < n; ++i) {
            long long x; cin>>x;
            mexo = max(mexo , x);
                sum += x;
        }
        long double res = mexo + ((sum-mexo ) /(n-1.0 ) ) ;
        std::cout << std::fixed;
        std::cout << std::setprecision(9);
        cout<<res<<endl;
    }
}
```

Problem H

In this problem any number Consists of { 1 , 14 , 144 } Only, print “YES”, Otherwise print “NO”.

1 1 4 1 1 4

All numbers { 1 , 14 , 144 } start with 1.

If S[i] == 1:

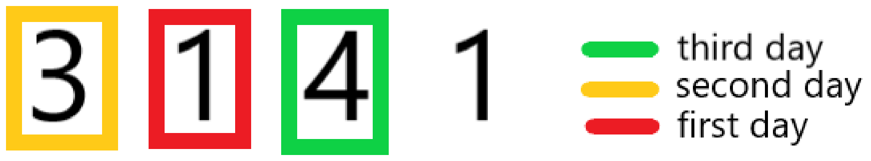
$$f(i) = \begin{cases} i+2, & S[i+1] == 4\ , S[i+2] == 4\ , i+2 < n \\ i+1, & S[i+1] == 4\ , i+1 < n \end{cases}$$

Otherwise, print NO.

```
#include<iostream>
using namespace std;
int main ()
{
string  s; cin>>s;
    for (int i = 0; i < s.size(); ++i) {
        if(s[i]=='1'){
            if(i+2<s.size() &&s[i+1]=='4' &&s[i+2]=='4'){i+=2;continue;}
            if(i+1<s.size() && s[i+1]=='4'){i++;continue;}
        }
        else
        {cout<<"NO"<<endl; return 0;}
    }
    cout<<"YES"<<endl;
    return 0;
}
```

Problem I

In this problem we need to maximize days he will train; he will stop training when he can’t get any contest have problems more than Day.



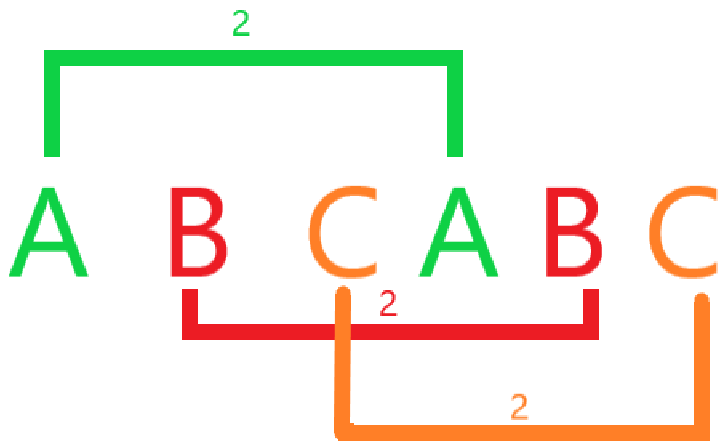
Day number	Number of problems	
1	1	$1 \geq 1$
2	3	$3 \geq 2$
3	4	$4 \geq 3$

So, I will create variable to simulate the day number, and to take optimally I must sort array, because without sorting may take greater value and miss mid value for example: (3 , 1 , 4 , 1) will take 3 in first day and 4 in second day.

```
#include<iostream>
using namespace std;
int main ()
{
    int n ; cin>>n ;
    int arr[n];
    in(arr,n);
    sort(arr , arr+n);
    int pos = 1 ;
    for (int i = 0; i < n; ++i) {
        if(arr[i] >= pos){pos ++;}
    }
    cout<< pos - 1 <<endl;}
```

Problem J

In this problem You are given a string *S* and every letter appears in it no more than twice, we need all pairs have the same distance:



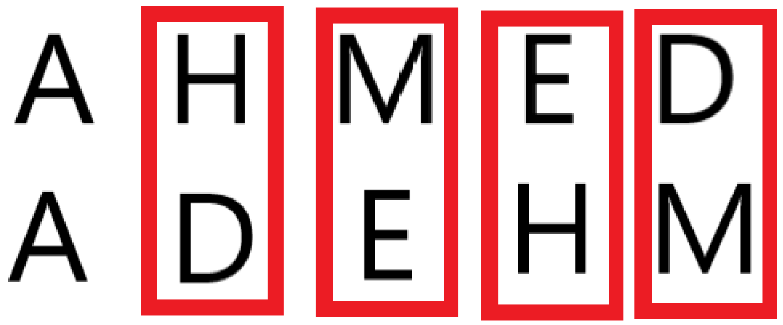
The distance between any pair is 2:

If we sort the string, the distance will be 1 between any two pairs.

```
#include<iostream>
using namespace std;
int main ()
{
int tc = 1 ;
    cin>>tc ;
    while (tc--)
    {
        string  s ;
        cin>>s;
        sort(s.begin() , s.end());
        cout<<s<<endl;
    }
}
```

Problem K

In this problem we need to sort our string with minimum number of characters.



To sort String “AHMED” I need to swap 4 characters.

```
#include<iostream>
using namespace std;
int main ()
{
int tc;
    cin>>tc;
    while(tc--)
    {
        string s;
        int size,ans=0;
        cin>>size>>s;
        string n=s;
        sort(s.begin(),s.end());
        for(int i=0 ; i<size ; i++)
        {
            if(n[i]!=s[i])
            {
                ans++;
            }
        }
        cout<<ans<<endl;
    }
    return 0;
}
```