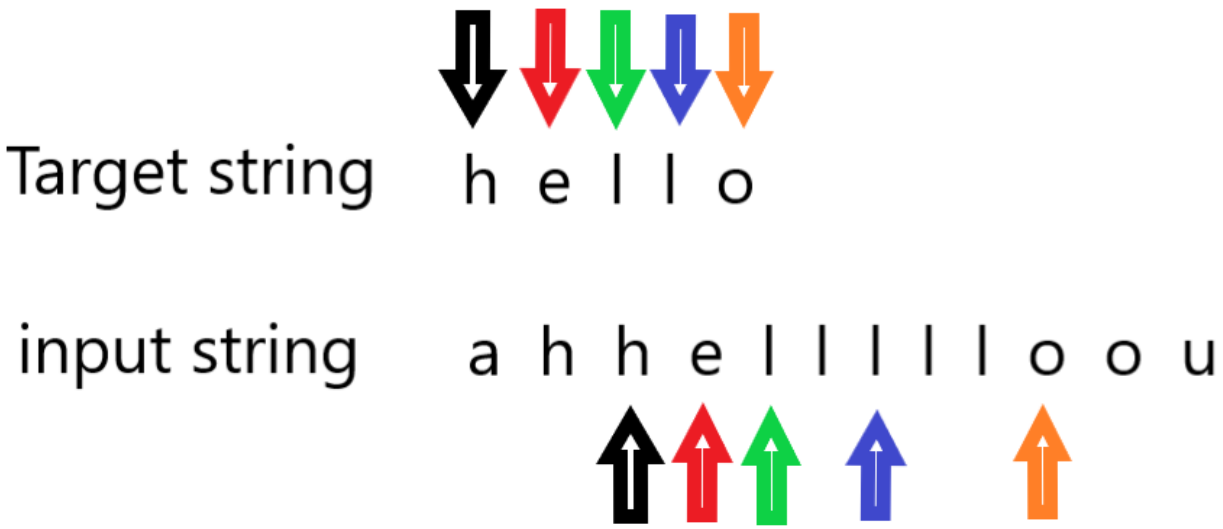
Problem A

We need to find string: "*hello*" in input string, so we will make pointer in *current_character* in "*hello*" and *search_character* in input string.



So, when *search_character* = *current_character* we increment *current_character*, otherwise increment *search_character*.

```
int main() {
    string trg ="hello",in;
    int j=0;
    cin>>in;
    for(char i : in)
    {
        if(i==trg[j]){j++;}
    }
    if(j==5){cout<<"YES"<<endl;}
    else{cout<<"NO"<<endl;}
}
```

Problem B

We can represent data of laptops as such:

First	Price
Second	Quality

So we need to use array pairs.

Alex search for two laptops X and Y such that

$$(X.Price < Y.Price \text{ and } X.Quality > Y.Quality)$$

At first, we will sort laptops on Price, it guarantees that $Arr[i+1] \geq Arr[i]$

Now we must check if Condition is met or not.

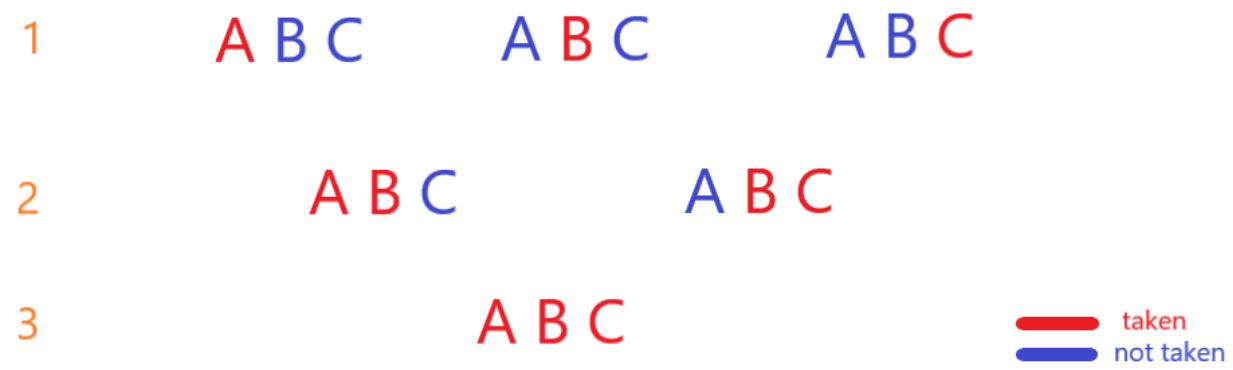
```
int main() {
    int n ; cin>>n;
    pair<int,int> labtops [n] ;
    for (int i = 0; i < n; ++i) {
        cin>>labtops[i].first>>labtops[i].second;
    }
    sort(labtops,labtops+n);
    for (int i = 0; i < n; ++i) {
        if(labtops[i].first < labtops[i+1].first &&
labtops[i].second>labtops[i+1].second)
        {
            cout<<"Happy Alex"<<endl;
            return 0;
        }
    }
    cout<<"Poor Alex"<<endl;
    return 0;
}
```

Problem C

Number of substrings is

$$\frac{N\,(N\,+\,1\,)}{2}$$

Proof:



If N is number of distinct characters in string. Answer is

$$\frac{N\,(N\,+\,1\,)}{2}$$

Problem D

In each row *I* if we find (1 , 3) it means this car will turned over during the collision.

In each column *J* if we find (2 , 3) it means this car will turned over during the collision.

So, we check for each row if this row dose not contain either 3 or 1 we will count this row.

And we Must use vector to push indices and to print its value.

```
int main() {
    int n  , x;
    cin>>n;
    vector<int> ans ;
    for (int i = 0; i < n; ++i) {
        bool check_row = false ;
        for (int j = 0; j <n ; ++j) {
            cin>>x;
            if(x==1 || x==3){check_row = true;}
        }
        if(!check_row){ans.emplace_back(i+1);}
    }
    cout<<ans.size()<<endl;
    for(int y : ans)
    {cout<<y<<" ";}
}
```

Problem E

$B \% C$ mean:

<u>Subtract how many numerators until it is divisible by the denominator</u>.

So, if $B \% C \leq B - a$ it means that we can subtract from numerator to be divisible by C.

Otherwise print -1.

```
void solve(int tc = 0) {
   int a , b , c ;
   cin>>a>>b>>c ;
   if(b%c <= (b-a)){cout<< b - (b%c) <<endl;}
   else{cout<<-1<<endl;}
}
```

Time complexity $O(1)$.

You can use loops and check each number if divisible by C in $O(B)$.

Problem F

This card gives you $a_i = 3$ points, and opportunity to play additional $b_i = 2$ cards.

Idea: try to play cards as much as possible to increase number of cards.

Firstly, we will sort all pairs based on b_i. it guarantees to take as much as possible cards and if two cards give same b_i will card that have a_i greater.

So, we will create pairs as:

points	second
cards	first

So, when we use sort it will sort pairs based on first, we need to take maximum element we can add another attribute greater<>() to sort function.

```
int main() {
    int n ; cin>>n;
    pair<int,int> B[n];
    int b=1 , a=0;
    for (int i = 0; i < n; ++i) {
        cin>>B[i].second>>B[i].first;
    }
    sort(B , B+n , greater<>());
    int sum = 0 , nof =1;
    for (int i = 0; i < n &&nof>0; ++i) {
        sum+=B[i].second;
        nof+= B[i].first -1;
    }
    cout<<sum<<endl;
}
```

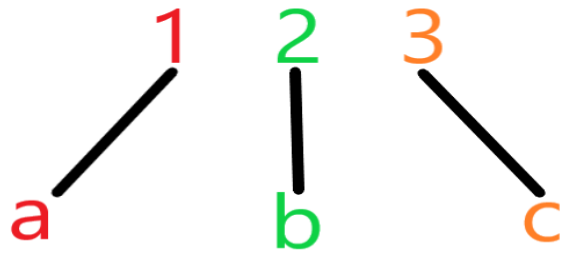
Problem G

we try to separate each digit in integer k.

Let's define first digit as c.

And second digit as b.

And third digit as a.



$$c = (k \% 10)$$

Now we have last digit in c , we need to get second digit.

So, we will divide by 10 because 123/10 = 12.3 in C++ we equal 12.

$$K = 12$$

Now we can get b by %10

$$b = (k \% 10)$$

Again, we will divide by 10 to get third digit

$$K = 1$$

Now we have last digit in K so we can say $a = k$ or

$$a = (k \% 10)$$

To get values back we multiply number in his weight:

$$abc = (a * 100) + (b * 10) + c$$

$$bca = (b * 100) + (c * 10) + a$$

$$cab = (c * 100) + (a * 10) + b$$

Problem J

Description	Current
Firstly, we have $K = 18$ let's swap with first element. $K = 81$ now	18 ⇩ 81 324 218 413 324
Here we have $K = 81$ and second element 324. Swap values, $K = 324$	81 ⇩ 18 324 218 413 324
Here we have $K = 324$ and fourth element 324. Swap values, $K = 413$	324 ⇩ 18 81 218 413 324
Now array is sorted and $K = 413$	413 ⇩ 18 81 218 324 324

We will use function is_sorted(arr , arr + n).

While array not sorted, we will iterate over array and check if we have any value $a_i > k$ we will swap these values.

If we didn't find any value $a_i > k$ and array not sorted it means we can't sort this array with value K so we will print -1

```
signed main()
{
    MOHARM
    int tc;
    cin>>tc;
    while (tc--)
    {
       int n , k ;
       cin>>n>>k ;
       int arr[n];
       in(arr,n)
       int ans = 0 ;
        while (!is_sorted(arr,arr+n))
        {
            bool not_sorted = true ;
            for (int &temp : arr) {
                if(temp>k){
                    swap(k ,temp );
                    not_sorted = false;
                    ++ans;
                    break;
                }
            }

//NO CHANGE AND WHILE CONDITION NOT TRUE -> NOT SORTED AND CAN'T UPDATE ANY ELEMENT
            if(not_sorted){ ans = - 1 ;break;}
        }
        cout<<ans<<endl;

    }
}
```

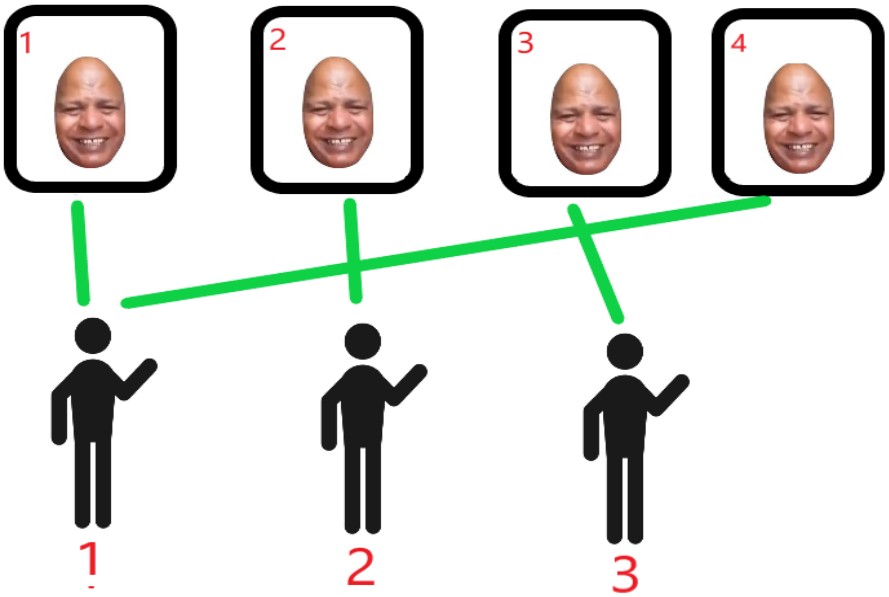
Problem K

In this problem he has A coin, and he want to buy item with B coin.

$$f(x)\begin{cases} 0 & ,(B-A) \geq 0 \\ (B-A)/10 & ,(B-A)\%10 = 0 \\ (B-A)/10+1 & ,(B-A)\%10 \neq 0 \end{cases}$$

```
void solve(int tc = 0) {
   int a , b ;
   cin>>a>>b ;
   int diff = (b - a) ;
   if(diff<=0){cout<<0;}
   else
   {
      cout<< ((b - a ) / 10) + (((b - a) % 10)  != 0)  ;
   }
}
```

Problem M



We need to share cards one by one on people.

Let's define number of cards A and number of people B.

If $B > A$ we will share cards to first A person.

If $B < A$ we will share cards to all persons and repeat this operation N time until we have no card.

<u>Mod : Subtract how many numerators until it is divisible by the denominator</u>.

Cards	People	note
3	3	Each one will take cards so, we will divide 3 cards for 3 persons and remaining 0 cards so answer 3 % 3 = 0 , so last person is 3
4	3	Each one will take cards so, we will divide 3 cards for 3 persons and remaining 0 cards so answer 4 % 3 = 1 , so last person is 1

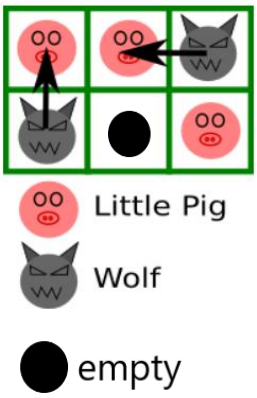
Because we start from A person so we will add this value to number of cards and subtract one because C will take card too.

$$(A + C - 1) \% B$$

Problem N

Let's create initial array with size 13 * 13 with empty value.

and fill values P and V from input.



- if current cell [i] [j] is W and [i] [$j + 1$] is P we will make [i] [$j + 1$] empty cell

- if current cell [i] [j] is W and [i] [$j - 1$] is P we will make[i] [$j - 1$] empty cell

- if current cell [i] [j] is W and [$i + 1$] [j] is P we will make [$i + 1$] [j] empty cell

- if current cell [i] [j] is W and [$i - 1$] [j] is P we will make [$i - 1$] [j] empty cell

```
void solve(int tc = 0) {
   cin>>n>>m ;
   char grid [ 12 ] [ 12 ] ;
//fill grid with .
    for (int i = 0; i <= 11; ++i) {
        for (int j = 0; j <= 11; ++j) {
            grid[i][j] = '.';
        }
    }
//take input
    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            cin>>grid[i][j];
        }
    }
//algorithm
    for (int i = 1; i <= 11; ++i) {
        for (int j = 1; j <= 11; ++j) {
            if(grid[i][j]=='W' )
            {
                if(grid[i][j+1]=='P') { grid[i][j + 1] = '.'; }
                else if (grid[i][j-1]=='P'){grid[i][j-1] = '.';}
                else if(grid[i+1][j]=='P'){grid[i+1][j] = '.';}
                else if(grid[i-1][j]=='P'){grid[i-1][j] = '.';}
                else{ans--;} //we will increase ans all times ( +1 - 1 )
                ans++;
            }
        }
    }
    cout<<ans<<endl;
}
```