<table>
<tr><th>Expt. No.</th><th>Objective of the Experiment</th></tr>
<tr><td>1</td><td>
Agastya is currently standing at stair 00 and he wants to reach stair numbered XX.
Agastya can climb either YY steps or 11 step in one move.
Find the minimum number of moves required by him to reach exactly the stair numbered XX.
Input Format
The first line of input will contain a single integer TT, denoting the number of test cases.
Each test case consists of a single line of input containing two space separated integers XX and YY denoting the number of stair Agastya wants to reach and the number of stairs he can climb in one move.
Output Format
For each test case, output the minimum number of moves required by him to reach exactly the stair numbered XX.
Constraints
1≤T≤500
1≤X,Y≤100
Sample:
<table>
<tr><th>Input</th><th>Output</th></tr>
<tr><td>4</td><td></td></tr>
<tr><td>4 2</td><td>2</td></tr>
<tr><td>8 3</td><td>4</td></tr>
<tr><td>3 4</td><td>3</td></tr>
<tr><td>2 1</td><td>2</td></tr>
</table>
Explanation:
Test case 1: Agastya can make 2 moves and climb 2 steps in each move to reach stair numbered 4.
Test case 2: Agastya can make a minimum of 4 moves. He can climb 3 steps in 2 of those moves and 1 step each in remaining 2 moves to reach stair numbered 8.
Test case 3: Agastya can make 3 moves and climb 1 step in each move to reach stair numbered 3.
Test case 4: Agastya can make 2 moves and climb 1 step in each move to reach stair numbered 2.


```
#include <iostream>
using namespace std;

int main() {
        // your code goes here
        int t,x,y,ans;
        cin>>t;
        while(t--)
```
</td></tr>
</table>

```
        {
           cin>>x>>y;
           ans=x/y;
           ans+=(x%y);
           cout<<ans<<endl;
        }

        return 0;
}
```

2

You are given the height H (in metres) and mass M (in kilograms). The Body Mass Index (BMI) of a person is computed as $\frac{M}{H^2}$.

Report the category into which a person falls, based on his BMI:

- Category 1: Underweight if $BMI \leq 18$
- Category 2: Normal weight if $BMI \in \{19, 20, \ldots, 24\}$
- Category 3: Overweight if $BMI \in \{25, 26, \ldots, 29\}$
- Category 4: Obesity if $BMI \geq 30$

Input:

The first line of input will contain an integer, TT, which denotes the number of testcases. Then the testcases follow.

Each testcase contains a single line of input, with two space separated integers, M, H, which denote the mass and height of Agastya respectively.

Output:

For each testcase, output in a single line, 1, 2, 31,2,3 or 44, based on the category in which Agastya falls.

Constraints

$1 \leq T \leq 2 * 10^4$

$1 \leq M \leq 10^4$

$1 \leq H \leq 10^2$

It is guaranteed that H^2 divides M.

Sample

Input	Output
3	
72 2	1
80 2	2
120 2	4

Explanation:

Case 1: Since $\frac{M}{H^2} = \frac{72}{2^2} = 18$, therefore person falls in category 1.

Case 2: Since $\frac{M}{H^2} = \frac{80}{2^2} = 20$, therefore person falls in category 2.

Case 3: Since $\frac{M}{H^2} = \frac{120}{2^2} = \ = 30$, therefore person falls in category 4.

```
#include <iostream>
using namespace std;

int main() {
        // your code goes here
```

```
        int t,m,h,b;
        cin >> t;
        while(t--){
           cin >> m>>h;
           b=m/(h*h);
           if(b<=18){
              cout << 1 << endl;
           }
           else if(b>=19 && b<=24){
        cout << 2 << endl;
           }
           else if(b>=25 && b<=29){
              cout << 3 << endl;
           }
           else{
              cout << 4 << endl;
           }

        }
        return 0;
}
```

You will be given an array of N integers and you have to print the integers in the reverse order.

Input Format

The first line of the input contains N, where N is the number of integers. The next line contains N space-separated integers.

Constraints

$1 \le N \le 1000$

$1 \le A[i] \le 10000$ where A[i] is the i^{th} integer in the array.

Sample:

Input	Output
4	
1 4 3 2	2 3 4 1

3

```
//We have populated the solutions for the 10 easiest problems for your support.
//Click on the SU[IT button to make a submission to this problem.

#include <bits/stdc++.h>
using namespace std;

int main() {
        int a, rem, sum = 0;
        cin>>a;
        while (a--) {

                int n;
                cin>>n;
                while(n!=0) {
                        rem = n % 10;
                        sum = sum * 10 + rem;
```

```
                n = n / 10;
            }
            cout<<sum<<endl;
            }
    return 0;

    }
```

4

Bhuvan has a string S with him. Bhuvan is happy if the string contains a contiguous substring of length strictly greater than 2 in which all its characters are vowels.
Determine whether Bhuvan is happy or not.
Note that, in English alphabet, vowels are a, e, i, o, and u.

Input Format

First line will contain T, number of test cases. Then the test cases follow.
Each test case contains of a single line of input, a string S.

Output Format

For each test case, if Bhuvan is happy, print HAPPY else print SAD.
You may print each character of the string in uppercase or lowercase (for example, the strings hAppY, Happy, haPpY, and HAPPY will all be treated as identical).

Constraints

$1 \leq T \leq 1000$
$3 \leq |S| \leq 1000$, where |S| is the length of S.
S will only contain lowercase English letters.

Sample:

Input	Output
4	
aeiou	Happy
abxy	Sad
aebcdefghij	Sad
abcdeeafg	Happy

Explanation:

Test case 1: Since the string aeiou is a contiguous substring and consists of all vowels and has a length >2, Agastya is happy.
Test case 2: Since none of the contiguous substrings of the string consist of all vowels and have a length >2, Agastya is sad.
Test case 3: Since none of the contiguous substrings of the string consist of all vowels and have a length >2, Agastya is sad.
Test case 4: Since the string eea is a contiguous substring and consists of all vowels and has a length >2, Agastya is happy.

4-

```
#include <iostream>
#include<string>
using namespace std;

int main()
{
        int t;
        cin>>t;
        while(t--)
        {
          string s;
          cin>>s;
          int j,i;
          int l=1;
          for(j=0;j<=s.length()-3;j++)
          {
             l=1;
             for(i=j;i<j+3;i++)
             {

                if(s[i]!='a' && s[i]!='e' && s[i]!='i' && s[i]!='o' && s[i]!='u')
                     {
                        l=0;
                     }
             }
             if(l==1)
               {
                cout<<"HAPPY"<<endl;break;}
          }
          if(l==0)
         cout<<"SAD"<<endl;
        }
        return 0;
}
```

5

There is a group of *N* friends who wish to enroll in a course together. The course has a maximum capacity of *M* students that can register for it. If there are *K* other students who have already enrolled in the course, determine if it will still be possible for all the *N* friends to do so or not.

Input Format

- The first line contains a single integer *T* - the number of test cases. Then the test cases follow.
- Each test case consists of a single line containing three integers *N*, *M* and *K* - the size of the friend group, the capacity of the course and the number of students already registered for the course.

Output Format

For each test case, output Yes if it will be possible for all the *N* friends to register for the course. Otherwise output No.

You may print each character of Yes and No in uppercase or lowercase (for example, yes, yEs, YES will be considered identical).

Constraints

- $1 \leq T \leq 1000$
- $1 \leq N \leq M \leq 100$
- $0 \leq K \leq M$

Sample

Input	Output
3	
2 50 27	Yes
5 40 38	No
100 100 0	Yes

```
#include <bits/stdc++.h>
using namespace std;

int main() {
        // your code goes here
        int t,N,M,K,c;
        cin>>t;
        while(t--)
        {
           cin>>N>>M>>K;
           c=K+N;
           if(c<=M)
           {
              cout<<"YES"<<endl;
           }
           else
           {
              cout<<"NO"<<endl;
           }
        }
        return 0;

}
```

Chetan is playing a videogame, and is getting close to the end. He decides to finish the rest of the game in a single session.

There are *X* levels remaining in the game, and each level takes Chetan *Y* minutes to complete. To protect against eye strain, Chetan also decides that every time he completes 3 levels, he will take a *Z* minute break from playing. Note that there is no need to take this break if the game has been completed.

How much time (in minutes) will it take Chetan to complete the game?

Input Format

- The first line of input will contain a single integer *T*, denoting the number of test cases.
- The first and only line of input will contain three space-separated integers *X*, *Y*, and *Z*.

Output Format

For each test case, output on a new line the answer — the length of Chetan's gaming session.

Constraints

- $1 \leq T \leq 100$
- $1 \leq X \leq 100$
- $5 \leq Y \leq 100$
- $5 \leq Z \leq 15$

Input	Output
4	
2 12 10	24
3 12 10	36
7 20 8	156
24 45 15	1185

6

```
#include <bits/stdc++.h>
using namespace std;

int main(void) {
        int t,x,y,z,sum,d,ld;
        cin>>t;
        while(t!=0){
           cin>>x>>y>>z;
           if(x<=3){
              sum=x*y;
           }else{
              ld=x%3;
              d=x/3;
              if(ld!=0){
                 sum=(ld*y)+(d*3*y)+(d*z);
              }else{
                 sum=(d*3*y)+((d-1)*z);
              }
           }
            cout<<sum<<endl;
            t--;
        }
        return 0;

}
```

Dhruv is very fond of horses. He enjoys watching them race. As expected, he has a stable full of horses. He, along with his friends, goes to his stable during the weekends to watch a few of these horses race. Agastya wants his friends to enjoy the race and so he wants the race to be close. This can happen only if the horses are comparable on their skill i.e. the difference in their skills is less.

There are **N** horses in the stable. The skill of the horse **i** is represented by an integer **S[i]**. Dhruv needs to pick 2 horses for the race such that the difference in their skills is *minimum*. This way, he would be able to host a very interesting race. Your task is to help him do this and report the minimum difference that is possible between 2 horses in the race.

Input:

First line of the input file contains a single integer **T**, the number of test cases. Every test case starts with a line containing the integer **N**. The next line contains **N** space separated integers where the **i**-th integer is **S[i]**.

Output:

For each test case, output a single line containing the minimum difference that is possible.

Constraints:

$1 \leq \mathbf{T} \leq 10$

$2 \leq \mathbf{N} \leq 5000$

$1 \leq \mathbf{S[i]} \leq 1000000000$

Sample 1:

Input	Output
1	
5	
4 9 1 32 13	3

Explanation:

The minimum difference can be achieved if we pick horses with skills 1 and 4 for the race.

7

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

int main() {
        // your code goes here
        int test;
        cin>>test;
        while(test)
        {
           int n;
           cin>>n;
           vector<long long int> v(n);
           for(int i=0;i<n;i++)
           {
              cin>>v[i];
           }

           sort(v.begin(),v.end());

          long long int min=INT_MAX;
           for(long long int i=0;i<n-1;i++)
           {
             long long int val=v[i+1]-v[i];
```

```
            if(min>val){min=val;}

        }
        cout<<min<<endl;
        test--;

    }

    return 0;
}
```

8

There is a frog initially placed at the origin of the coordinate plane. In exactly 1 second, the frog can either move up 1 unit, move right 1 unit, or stay still. In other words, from position (x,y), the frog can spend 1 second to move to:

- (x+1,y)
- (x,y+1)
- (x,y)

After T seconds, a villager who sees the frog reports that the frog lies on or inside a square of side-length s with coordinates (X,Y), (X+s,Y), (X,Y+s), (X+s,Y+s). Calculate how many points with integer coordinates on or inside this square could be the frog's position after exactly T seconds

Input Format:

The first and only line of input contains four space-separated integers: X, Y, s, and T.

Output Format:

Print the number of points with integer coordinates that could be the frog's position after T seconds.

Constraints:

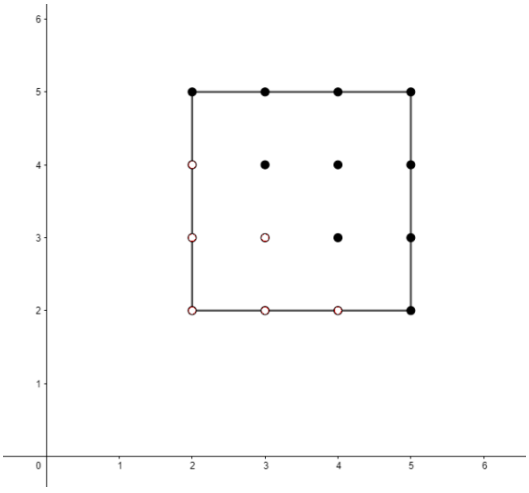
$0 \le X,Y \le 100$

$1 \le s \le 100$

$0 \le T \le 400$

Note that the Expected Output Feature for Custom Invocation is not supported for this contest.

Input	Output
2 2 3 6	6



The points shown are the points on or in the specified square, and those that are white are the points that the frog could be at after 6 seconds.

```
#include <iostream>
#include <bits/stdc++.h>
```

```
using namespace std;
int main() {

int x, y, s, t;
 cin >> x >> y >> s >> t;
 int p, d;
 // p is horizontal distance, d is the max distance horizontally or vertically
 p = t - (y - 1);

if(s + 1 < p) {
 d = s + 1;
 }
 else {
 d = p;
 }
 int count = 0;

for(int i = y; i <= y + d - 1; i++) {

if( (t + 1 - x - i) < (s + 1) && (t + 1 - x - i) > 0) {

count+= t + 1 - x - i;

}
 else if(t + 1 - x - i >= s + 1) {

 count+= s + 1;

}
 else {
 count+= 0;
 }

 }

cout << count;

 return 0;

}
```

9

There are *N* students in a class, where the *i*-th student has a score of *Ai*.
The *i*-th student will *boast* if and only if the number of students scoring less than or equal to *Ai* is greater than the number of students scoring greater than *Ai*.
Find the number of students who will boast.

Input Format

- The first line contains *T* - the number of test cases. Then the test cases follow.
- The first line of each test case contains a single integer *N* - the number of students.
- The second line of each test case contains *N* integers 1,2,…,*A*1,*A*2,…,*AN* - the scores of the students.

Output Format

For each test case, output in a single line the number of students who will boast.

Constraints

- $1 \leq T \leq 1000$
- $1 \leq N \leq 100$
- $0 \leq Ai \leq 100$

Input	Output
3	
3	
100 100 100	3
3	
2 1 3	2
4	
30 1 30 30	3

Explanation:
Test case1: All three students got the highest score. So all three of them will boast.
Test case2: Only the student with score 1 will not be able to boast.
Test case 3: Only the student with score 1 will not be able to boast.

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

int main() {
	// your code goes here
	int t;
	cin>>t;
	while(t--){
	   int n,current=0;
	   cin>>n;
	   int a[n];
	   for(int i=0;i<n;i++){
	      cin>>a[i];
	   }
	   sort(a,a+n);
	   for(int i=n/2;i>=0;i--){
	      if(a[i] != a[n/2]){
	         current=i+1;
	         break;
	      }
	   }
	   cout<<n-current<<endl;
	}
	return 0;
}
```