Module 10.5 Problem-set

<https://vjudge.net/contest/491097>

**#module\_release #course\_2 #module\_10.5**

কন্টেস্ট লিংকঃ <https://vjudge.net/contest/491097>

পাসওয়ার্ডঃ practicepracticepractice

## [A - Curtain](https://vjudge.net/problem/AtCoder-abc143_a)

[AtCoder - abc143\_a](https://vjudge.net/problem/AtCoder-abc143_a/origin" \t "_blank)

**Problem Statement**

The window of Takahashi's room has a width of *AA*. There are two curtains hung over the window, each of which has a horizontal length of *BB*. (Vertically, the curtains are long enough to cover the whole window.)

We will close the window so as to minimize the total horizontal length of the uncovered part of the window. Find the total horizontal length of the uncovered parts of the window then.

**Constraints**

* *1 \leq A \leq 1001≤A≤100*
* *1 \leq B \leq 1001≤B≤100*
* *AA* and *BB* are integers.

**Input**

Input is given from Standard Input in the following format:

*AA* *BB*

**Output**

Print the total horizontal length of the uncovered parts of the window.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 12 4 | 4 |

We have a window with a horizontal length of *1212*, and two curtains, each of length *44*, that cover both ends of the window, for example. The uncovered part has a horizontal length of *44*.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 20 15 | 0 |

If the window is completely covered, print *00*.

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 20 30 | 0 |

Each curtain may be longer than the window.

#include<stdio.h>

int main()

{

int a, b, c;

scanf("%d %d", &a, &b);

c=a-2\*b;

if(c>0)

{

printf("%d",c);

}

else

{

printf("0");

}

return 0;

}

## [B - Not Overflow](https://vjudge.net/problem/AtCoder-abc237_a)

[AtCoder - abc237\_a](https://vjudge.net/problem/AtCoder-abc237_a/origin" \t "_blank)

**Problem Statement**

You are given an integer *NN*. If *NN* is between *-2^{31}−231* and *2^{31}-1231−1* (inclusive), print Yes; otherwise, print No.

**Constraints**

* *-2^{63} \leq N < 2^{63}−263≤N<263*
* *NN* is an integer.

**Input**

Input is given from Standard Input in the following format:

*NN*

**Output**

If *NN* is between *-2^{31}−231* and *2^{31}-1231−1* (inclusive), print Yes; otherwise, print No.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 10 | Yes |

*1010* is between *-2^{31}−231* and *2^{31}-1231−1*, so Yes should be printed.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| -9876543210 | No |

*-9876543210−9876543210* is less than *-2^{31}−231*, so No should be printed.

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 483597848400000 | No |

*483597848400000483597848400000* is greater than *2^{31}-1231−1*, so No should be printed.

#include<stdio.h>

#include<math.h>

int main()

{

long long n;

scanf("%lld", &n);

if(n>=pow(-2,31) && n<=(pow(2,31)-1))

{

printf("Yes");

}

else

{

printf("No");

}

return 0;

}

## [C - Failing Grade](https://vjudge.net/problem/AtCoder-abc222_b)

[AtCoder - abc222\_b](https://vjudge.net/problem/AtCoder-abc222_b/origin" \t "_blank)

**Problem Statement**

*NN* students took an exam. The students are labeled as Student *11*, Student *22*, *\dots…*, Student *NN*, and Student *ii* scored *a\_iai​* points.

A student who scored less than *PP* points are considered to have failed the exam and cannot earn the credit. Find the number of students who failed the exam.

**Constraints**

* *1 \leq N \leq 10^51≤N≤105*
* *1 \leq P \leq 1001≤P≤100*
* *0 \leq a\_i \leq 1000≤ai​≤100* *(1 \leq i \leq N)(1≤i≤N)*
* All values in input are integers.

**Input**

Input is given from Standard Input in the following format:

*NN* *PP*

*a\_1a1​* *a\_2a2​* *\dots…* *a\_NaN​*

**Output**

Print the number of students who failed the exam.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 4 50  80 60 40 0 | 2 |

Students *11* and *22*, who scored *8080* and *6060* points, respectively, succeeded in scoring at least *5050* points to earn the credit.  
On the other hand, Students *33* and *44*, who scored *4040* and *00* points, respectively, fell below *5050* points and failed the exam. Thus, the answer is *22*.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3 90  89 89 89 | 3 |

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 2 22  6 37 | 1 |

#include<stdio.h>

int main()

{

int n, p, score[100001], count=0;

scanf("%d %d", &n, &p);

for(int i=1; i<=n; i++)

{

scanf("%d", &score[i]);

}

for(int i=1; i<=n; i++)

{

if(score[i]<p)

{

count++;

}

}

printf("%d", count);

return 0;

}

## [D - Shampoo](https://vjudge.net/problem/AtCoder-abc243_a)

[AtCoder - abc243\_a](https://vjudge.net/problem/AtCoder-abc243_a/origin" \t "_blank)

**Problem Statement**

Three people live in Takahashi's house: Takahashi, his father, and his mother. All of them wash their hair in the bathroom each night.  
His father, his mother, and Takahashi take a bath in this order and use *AA*, *BB*, and *CC* milliliters of shampoo, respectively.

This morning, the bottle contained *VV* milliliters of shampoo. Without refilling, who will be the first to run short of shampoo to wash their hair?

**Constraints**

* *1 \leq V,A,B,C \leq 10^51≤V,A,B,C≤105*
* All values in input are integers.

**Input**

Input is given from Standard Input in the following format:

*VV* *AA* *BB* *CC*

**Output**

If the first person to run short of shampoo to wash their hair is Takahashi's father, print F; if it is Takahashi's mother, print M; if it is Takahashi, print T.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 25 10 11 12 | T |

Now, they have *2525* milliliters of shampoo.

* First, Takahashi's father uses *1010* milliliters, leaving *1515*.
* Next, Takahashi's mother uses *1111* milliliters, leaving *44*.
* Finally, Takahashi tries to use *1212* milliliters and runs short of shampoo since only *44* is remaining.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 30 10 10 10 | F |

Now, they have *3030* milliliters of shampoo.

* First, Takahashi's father uses *1010* milliliters, leaving *2020*.
* Next, Takahashi's mother uses *1010* milliliters, leaving *1010*.
* Then, Takahashi uses *1010* milliliters, leaving *00*.
* Next day, Takahashi's father tries to use *1010* milliliters and runs short of shampoo since only *00* is remaining.

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 100000 1 1 1 | M |

#include<stdio.h>

int main()

{

int v, a, b, c;

scanf("%d %d %d %d", &v, &a, &b, &c);

if(v>=(a+b+c))

{

v=v%(a+b+c);

}

if(v<a)

{

printf("F");

}

else if(v<(a+b))

{

printf("M");

}

else if(v<(a+b+c))

{

printf("T");

}

return 0;

}

## [E - Rolling Dice](https://vjudge.net/problem/AtCoder-abc208_a)

[AtCoder - abc208\_a](https://vjudge.net/problem/AtCoder-abc208_a/origin" \t "_blank)

**Problem Statement**

Is it possible to get a sum of *BB* when throwing a die with six faces *1,2,\ldots,61,2,…,6* *AA* times?

**Constraints**

* *1 \leq A \leq 1001≤A≤100*
* *1 \leq B \leq 10001≤B≤1000*
* *AA* and *BB* are integers.

**Input**

Input is given from Standard Input in the following format:

*AA* *BB*

**Output**

If it is possible to get a sum of *BB*, print Yes; otherwise, print No.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 2 11 | Yes |

There are two ways to get a sum of *1111* when throwing a *66*-faced die twice:

* getting *66* in the first throw and *55* in the second throw;
* getting *55* in the first throw and *66* in the second throw.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 2 13 | No |

There is no way to get a sum of *1313* when throwing a *66*-faced die twice.

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 100 600 | Yes |

#include<stdio.h>

int main()

{

int a, b;

scanf("%d %d", &a, &b);

if(a\*6>=b && a<=b)

{

printf("Yes");

}

else

{

printf("No");

}

return 0;

}

## [F - Seismic magnitude scales](https://vjudge.net/problem/AtCoder-abc221_a)

[AtCoder - abc221\_a](https://vjudge.net/problem/AtCoder-abc221_a/origin" \t "_blank)

**Problem Statement**

The magnitude of an earthquake is a logarithmic scale of the energy released by the earthquake. It is known that each time the magnitude increases by *11*, the amount of energy gets multiplied by approximately *3232*.  
Here, we assume that the amount of energy gets multiplied by exactly *3232* each time the magnitude increases by *11*. In this case, how many times is the amount of energy of a magnitude *AA* earthquake as much as that of a magnitude *BB* earthquake?

**Constraints**

* *3\leq B\leq A\leq 93≤B≤A≤9*
* *AA* and *BB* are integers.

**Input**

Input is given from Standard Input in the following format:

*AA* *BB*

**Output**

Print the answer as an integer.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 6 4 | 1024 |

*66* is *22* greater than *44*, so a magnitude *66* earthquake has *32\times 32=102432×32=1024* times as much energy as a magnitude *44* earthquake has.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 5 5 | 1 |

Earthquakes with the same magnitude have the same amount of energy.

#include<stdio.h>

#include<math.h>

int main()

{

int a, b, m, p;

scanf("%d %d", &a, &b);

m=a-b;

p=pow(32,m);

printf("%lld", p);

return 0;

}

## [G - Century](https://vjudge.net/problem/AtCoder-abc200_a)

[AtCoder - abc200\_a](https://vjudge.net/problem/AtCoder-abc200_a/origin" \t "_blank)

**Problem Statement**

In what century is the year *NN*?

What is century?

**Constraints**

* *1 \le N \le 30001≤N≤3000*

**Input**

Input is given from Standard Input in the following format:

*NN*

**Output**

Print the answer as an integer.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 2021 | 21 |

This year *20212021* is in the *2121*-st century.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 200 | 2 |

The year *200200* is in the *22*-nd century.

#include<stdio.h>

#include<math.h>

int main()

{

int n, c;

scanf("%d", &n);

c=ceil((float)n/100);

printf("%d", c);

return 0;

}

## [H - Vanya and Fence](https://vjudge.net/problem/CodeForces-677A)

[CodeForces - 677A](https://vjudge.net/problem/CodeForces-677A/origin" \t "_blank)

Vanya and his friends are walking along the fence of height *h* and they do not want the guard to notice them. In order to achieve this the height of each of the friends should not exceed *h*. If the height of some person is greater than *h* he can bend down and then he surely won't be noticed by the guard. The height of the *i*-th person is equal to *ai*.

Consider the width of the person walking as usual to be equal to 1, while the width of the bent person is equal to 2. Friends want to talk to each other while walking, so they would like to walk in a single row. What is the minimum width of the road, such that friends can walk in a row and remain unattended by the guard?

**Input**

The first line of the input contains two integers *n* and *h* (1 ≤ *n* ≤ 1000, 1 ≤ *h* ≤ 1000) — the number of friends and the height of the fence, respectively.

The second line contains *n* integers *ai* (1 ≤ *ai* ≤ 2*h*), the *i*-th of them is equal to the height of the *i*-th person.

**Output**

Print a single integer — the minimum possible valid width of the road.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3 7  4 5 14 | 4 |

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 6 1  1 1 1 1 1 1 | 6 |

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 6 5  7 6 8 9 10 5 | 11 |

**Note**

In the first sample, only person number 3 must bend down, so the required width is equal to 1 + 1 + 2 = 4.

In the second sample, all friends are short enough and no one has to bend, so the width 1 + 1 + 1 + 1 + 1 + 1 = 6 is enough.

In the third sample, all the persons have to bend, except the last one. The required minimum width of the road is equal to 2 + 2 + 2 + 2 + 2 + 1 = 11.

#include<stdio.h>

#include<math.h>

int main()

{

int n, h, w, sum=0;

scanf("%d %d", &n, &h);

int a[1100], i;

for(i=1; i<=n; i++)

{

scanf("%d", &a[i]);

}

for(i=1; i<=n; i++)

{

if(a[i]<=h)

{

w=1;

}

else

{

w=2;

}

sum=sum+w;

}

printf("%d", sum);

return 0;

}

## [I - Drinks](https://vjudge.net/problem/CodeForces-200B)

[CodeForces - 200B](https://vjudge.net/problem/CodeForces-200B/origin" \t "_blank)

Little Vasya loves orange juice very much. That's why any food and drink in his kitchen necessarily contains orange juice. There are *n* drinks in his fridge, the volume fraction of orange juice in the *i*-th drink equals *pi* percent.

One day Vasya decided to make himself an orange cocktail. He took equal proportions of each of the *n* drinks and mixed them. Then he wondered, how much orange juice the cocktail has.

Find the volume fraction of orange juice in the final drink.

**Input**

The first input line contains a single integer *n* (1 ≤ *n* ≤ 100) — the number of orange-containing drinks in Vasya's fridge. The second line contains *n* integers *pi* (0 ≤ *pi* ≤ 100) — the volume fraction of orange juice in the *i*-th drink, in percent. The numbers are separated by a space.

**Output**

Print the volume fraction in percent of orange juice in Vasya's cocktail. The answer will be considered correct if the absolute or relative error does not exceed 10 - 4.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3  50 50 100 | 66.666666666667 |

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 4  0 25 50 75 | 37.500000000000 |

**Note**

Note to the first sample: let's assume that Vasya takes *x* milliliters of each drink from the fridge. Then the volume of pure juice in the cocktail will equal https://vj.csgrandeur.cn/13cab0fe37ccf2b27b15f713a9274a23?v=1650897245 milliliters. The total cocktail's volume equals 3·*x* milliliters, so the volume fraction of the juice in the cocktail equals https://vj.csgrandeur.cn/358afb131ffb84e098ca5cc25d76ef6e?v=1650897245, that is, 66.(6) percent.

#include<stdio.h>

int main()

{

int n;

float sum=0;

scanf("%d",&n);

for(int i=1;i<=n;i++)

{

int a;

scanf("%d",&a);

sum+=a;

}

printf("%f",sum/n);

}

## [J - Round decimals](https://vjudge.net/problem/AtCoder-abc226_a)

[AtCoder - abc226\_a](https://vjudge.net/problem/AtCoder-abc226_a/origin" \t "_blank)

**Problem Statement**

You are given a real number *XX*, which is representable using at most three decimal digits, with three decimal digits.  
Round *XX* to the nearest integer and print the result.

**Constraints**

* *0 \leq X < 1000≤X<100*
* *XX* is representable using at most three decimal digits.
* *XX* has three decimal digits in input.

**Input**

Input is given from Standard Input in the following format:

*XX*

**Output**

Print the integer resulting from rounding *XX* to the nearest integer.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3.456 | 3 |

The digit in the first decimal place of *3.4563.456* is *44*, so we should round it down to *33*.

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 99.500 | 100 |

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 0.000 | 0 |

#include<stdio.h>

int main()

{

float n, r;

scanf("%f", &n);

int i, p;

i=(int)n;

r=n-i;

if(r>=0.5)

{

p=ceil(n);

}

else

{

p=floor(n);

}

printf("%d", p);

return 0;

}

## [K - In Search of an Easy Problem](https://vjudge.net/problem/CodeForces-1030A)

[CodeForces - 1030A](https://vjudge.net/problem/CodeForces-1030A/origin" \t "_blank)

When preparing a tournament, Codeforces coordinators try treir best to make the first problem as easy as possible. This time the coordinator had chosen some problem and asked n*n* people about their opinions. Each person answered whether this problem is easy or hard.

If at least one of these n*n* people has answered that the problem is hard, the coordinator decides to change the problem. For the given responses, check if the problem is easy enough.

**Input**

The first line contains a single integer n*n* (1 \le n \le 1001≤*n*≤100) — the number of people who were asked to give their opinions.

The second line contains n*n* integers, each integer is either 00 or 11. If i*i*-th integer is 00, then i*i*-th person thinks that the problem is easy; if it is 11, then i*i*-th person thinks that the problem is hard.

**Output**

Print one word: "EASY" if the problem is easy according to all responses, or "HARD" if there is at least one person who thinks the problem is hard.

You may print every letter in any register: "EASY", "easy", "EaSY" and "eAsY" all will be processed correctly.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3  0 0 1 | HARD |

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 1  0 | EASY |

**Note**

In the first example the third person says it's a hard problem, so it should be replaced.

In the second example the problem easy for the only person, so it doesn't have to be replaced.

#include<stdio.h>

int main()

{

int n, i, a[100], count=0;

scanf("%d", &n);

for(i=1; i<=n; i++)

{

scanf("%d", &a[i]);

}

for(i=1; i<=n; i++)

{

if(a[i]==1)

{

count++;

break;

}

}

if(count!=0)

{

printf("HARD");

}

else

{

printf("EASY");

}

return 0;

}

## [L - Dislike of Threes](https://vjudge.net/problem/CodeForces-1560A)

[CodeForces - 1560A](https://vjudge.net/problem/CodeForces-1560A/origin" \t "_blank)

Polycarp doesn't like integers that are divisible by 33 or end with the digit 33 in their decimal representation. Integers that meet both conditions are disliked by Polycarp, too.

Polycarp starts to write out the positive (greater than 00) integers which he likes: 1, 2, 4, 5, 7, 8, 10, 11, 14, 16, \dots1,2,4,5,7,8,10,11,14,16,…. Output the k*k*-th element of this sequence (the elements are numbered from 11).

**Input**

The first line contains one integer t*t* (1 \le t \le 1001≤*t*≤100) — the number of test cases. Then t*t* test cases follow.

Each test case consists of one line containing one integer k*k* (1 \le k \le 10001≤*k*≤1000).

**Output**

For each test case, output in a separate line one integer x*x* — the k*k*-th element of the sequence that was written out by Polycarp.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 10  1  2  3  4  5  6  7  8  9  1000 | 1  2  4  5  7  8  10  11  14  1666 |

#include<stdio.h>

int main()

{

int t;

scanf("%d", &t);

for(int i=1; i<=t; i++)

{

int k, last=0, count=0;

scanf("%d", &k);

for(int j=1; count<k; j++)

{

if(j%3==0 || j%10==3)

{

continue;

}

else

{

count++;

last=j;

}

}

printf("%d\n", last);

}

return 0;

}

## [M - New Generation ABC](https://vjudge.net/problem/AtCoder-abc214_a" \t "_blank)

[AtCoder - abc214\_a](https://vjudge.net/problem/AtCoder-abc214_a/origin" \t "_blank)

**Problem Statement**

This is the *214214*-th AtCoder Beginner Contest (ABC).

The ABCs so far have had the following number of problems.

* The *11*-st through *125125*-th ABCs had *44* problems each.
* The *126126*-th through *211211*-th ABCs had *66* problems each.
* The *212212*-th through *214214*-th ABCs have *88* problems each.

Find the number of problems in the *NN*-th ABC.

**Constraints**

* *1 \leq N \leq 2141≤N≤214*
* All values in input are integers.

**Input**

Input is given from Standard Input in the following format:

*NN*

**Output**

Print the answer.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 214 | 8 |

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 1 | 4 |

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 126 | 6 |

#include<stdio.h>

int main()

{

int n;

scanf("%d", &n);

if(n>=1 && n<=125)

{

printf("4");

}

else if(n>=126 && n<=211)

{

printf("6");

}

else if(n>=212 && n<=214)

{

printf("8");

}

return 0;

}

## [N - Police Recruits](https://vjudge.net/problem/CodeForces-427A)

[CodeForces - 427A](https://vjudge.net/problem/CodeForces-427A/origin" \t "_blank)

The police department of your city has just started its journey. Initially, they don’t have any manpower. So, they started hiring new recruits in groups.

Meanwhile, crimes keeps occurring within the city. One member of the police force can investigate only one crime during his/her lifetime.

If there is no police officer free (isn't busy with crime) during the occurrence of a crime, it will go untreated.

Given the chronological order of crime occurrences and recruit hirings, find the number of crimes which will go untreated.

**Input**

The first line of input will contain an integer *n* (1 ≤ *n* ≤ 105), the number of events. The next line will contain *n* space-separated integers.

If the integer is -1 then it means a crime has occurred. Otherwise, the integer will be positive, the number of officers recruited together at that time. No more than 10 officers will be recruited at a time.

**Output**

Print a single integer, the number of crimes which will go untreated.

**Sample 1**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 3  -1 -1 1 | 2 |

**Sample 2**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 8  1 -1 1 -1 -1 1 1 1 | 1 |

**Sample 3**

| **Inputcopy** | **Outputcopy** |
| --- | --- |
| 11  -1 -1 2 -1 -1 -1 -1 -1 -1 -1 -1 | 8 |

**Note**

Lets consider the second example:

1. Firstly one person is hired.
2. Then crime appears, the last hired person will investigate this crime.
3. One more person is hired.
4. One more crime appears, the last hired person will investigate this crime.
5. Crime appears. There is no free policeman at the time, so this crime will go untreated.
6. One more person is hired.
7. One more person is hired.
8. One more person is hired.

The answer is one, as one crime (on step 5) will go untreated.

#include <stdio.h>

int main()

{

int n,a,uc=0,sum=0;

scanf("%d", &n);

for(int i=1; i<=n; i++)

{

scanf("%d", &a);

if(a==-1)

{

if(sum>0)

{

sum=sum+a;

}

else

{

uc++;

}

}

else

{

sum=sum+a;

}

}

printf("%d", uc);

return 0;

}