Хмельницький обласний інститут післядипломної
педагогічної освіти імені Анатолія Назаренка

Сергій ВАПНІЧНИЙ
Віталій ЗУБИК
Віталій РЕБРИНА

99 ЗАДАЧ
ІЗ РОЗВ'ЯЗКАМИ МОВОЮ ПРОГРАМУВАННЯ

Хмельницький
2024

УДК 004.43 (078)
В - 12

Рецензенти:
О.С. Сологуб, старший викладач кафедри психології та інклюзивної освіти, методик природничо-математичних дисциплін і технологій Хмельницького обласного інституту післядипломної педагогічної освіти ім. А. Назаренка, доктор філософії
О.Д. Куліковський, учитель інформатики комунального закладу загальної середньої освіти «Ліцей № 3 імені Артема Мазура Хмельницької міської ради», учитель-методист

В - 12 Вапнічний С.Д., В.В.Зубик, Ребрина В.А.
99 задач із розв’язками мовою програмування С++/ Вапнічний С.Д., Зубик В.В., Ребрина В.А.: [Навч.посіб.]. – Хмельницький – 2024 .-107 с.
УДК 004.43 (078)

У посібнику розглянуті 99 задач із сайту «Школа олімпійського резерву» [доступ https://sbs.hoippo.km.ua/] і надані розв’язки цих задач на мові програмування С++. Сайт ідеально підходить для початкового вивчення основ програмування та алгоритмізації. На сервері перевірки всі завдання розбиті на теми та групи - можна швидко підібрати завдання відповідно до потреби.

Посібник розрахований на вчителів інформатики та учнів, які займаються вивченням сучасних мов програмування.

Затверджено вченою радою Хмельницького обласного інституту післядипломної педагогічної освіти імені Анатолія Назаренка, протокол № 1 від 27.06.2024 р.

Лінійні програми

Задача 1 (1000: Площа)

Дано цілі числа **a, b** - катети прямокутного трикутника. Напишіть програму, яка знаходить площу прямокутного трикутника.

Формат вхідних даних.

Вхідний потік містить натуральні числа **a, b** не більші 10^6. Числа розділяються пропуском.

Формат вихідних даних:

Вивести площу з двома знаками після коми.

Приклад вхідних даних:

```
4 5
```

Приклад вихідних даних:

```
10.00
```

Програма мовою C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  int a, b;
  cin >> a >> b;
  double s = a * b / 2.0;
  cout << fixed << setprecision(2) << s;
}
```

Пояснення: формула для площі трикутника $S = \frac{ab}{2}$. У **C++** діє правило: *ціле число поділити на ціле результат теж число ціле*. Щоб результат отримати типу **double** необхідно зробити наступне: або у формулу внести/перетворити один операнд у тип **double** або весь вираз примусово перетворити у дійсний тип, наприклад:

```
  double s = (double) a * b / 2;
```

Підключення бібліотеки **<iomanip>** необхідне для роботи маніпуляторів **fixed** та **setprecision(2)** – які забазпечують вивід відповіді з двома знаками після коми.

==========

Задача 2 (1001: Сума дійсних)

Дано чотири дійсні числа **a, b, c, d.** Знайти їх суму.

Формат вхідних даних.

Вхідний потік містить дійсні числа **a, b, c, d.** Числа розділяються пропуском та по модулю не більші 10^9

Формат вихідних даних:

Вивести результат з чотирма знаками після коми.

Приклад вхідних даних::

```
3  4  2.5  1
```

Приклад вихідних даних::

```
10.5000
```

Програма мовою C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
    {
        double a, b, c, d;
        cin >> a >> b >> c >> d;
        double s = a + b + c + d;
        cout << fixed << setprecision(4) << s;
    }
```

==========

Задача 3 (1002: Периметр)

Знайти периметр прямокутника, якщо відомо дві його сторони ***a*** та ***b*.**

Формат вхідних даних..

Вхідний потік містить цілі числа ***a, b*** **(1 < *a, b* < 10^9)** , які розділяються пропуском

Формат вихідних даних.

Вивести периметр.

Приклад вхідних даних::

```
2 6
```

Приклад вхідних даних::

```
16
```

Програма мовою С++:

```
#include <iostream>
using namespace std;
int main()
{
  long long a, b;
  cin >> a >> b;
  long long p = (a + b) * 2;
  cout << p;
}
```

Пояснення: Якщо вхідні дані **a, b** описати як тип **int** , то у формулі для обчислення периметра слід виконати наступне: або один операнд зробити типу **long long** (константу взяти типу **long long**)отримаємо:

```
long long p = (a + b) * 2LL;
```

або примусово перетворити результат у тип **long long**:

```
long long p = (long long)(a + b) * 2;
```

==========

Задача 4 (1003: Остача)

Дано цілі числа ***x, y***. В першому рядку вивести остачу від ділення першого числа на друге, а в другому остачу від ділення другого на перше.

Формат вхідних даних..

Вхідний потік містить цілі числа **x, y (0 < x, y < 10^9)**. Числа розділяються пропуском.

Формат вихідних даних.

В першому рядку вивести остачу від ділення першого числа на друге, а в другому остачу від ділення другого на перше.

Приклад вхідних даних::

```
15 16
```

Приклад вихідних даних::

```
15
```

```
1
```

Програма мовою С++:

```
#include <iostream>
using namespace std;
int main()
{
  int a, b;
  cin >> a >> b ;
  cout << a % b << '\n' << b % a;;
}
```

Пояснення: перехід на новий рядок (клік на клавішу ентер) у цій програмі і в подальшому подано у С-варіанті: `'\n'`. Можна використовувати і `endl` .

==========

Задача 5 (1004: Дійсне на частини)

Дано дійсне число **p**. В першому рядку вивести його заокруглене ціле значення, в другому - його цілу частину, в третьому - його дробову частину (один знак після коми).

Формат вхідних даних..

Вхідний потік містить додатне дійсне число $p (p < 10^{12})$

Формат вихідних даних.

В першому рядку вивести його заокруглене ціле значення, в другому - його цілу частину, в третьому - його дробову частину (один знак після коми).

Приклад вхідних даних::

```
3.5
```

Приклад вихідних даних::

```
4
3
0.5
```

Програма мовою С++:

```
#include <iostream>
#include <iomanip>
```

```
using namespace std;
int main()
{
   double x;
   cin >> x;
   int a = (int)x;         // ціла частина
   int b = (int)(x+0.5); // округлене число
   double c = x - a;       // дробова частина
   cout << b << '\n' << a << '\n' << fixed <<
setprecision(1) << c;
}
```

Пояснення: округлення вверх та вниз у цій програмі виконано без використання математичних функцій **floor()** та **ceil()** які описані в бібліотеці **<cmath>**

==========

Задача 6 (1005: Значення виразу)

Дано цілі числа ***x*** та ***y***. Знайти значення виразу: $x^3+\frac{x+1}{y^2+1}$

Формат вхідних даних..

Стандартний потік містить цілі числа ***x***, ***y*** (**0 < *x*, *y* < 10^9**).

Числа розділяються пропуском.

Формат вихідних даних.

В результаті вивести один знак після коми.

Приклад вхідних даних::

```
3 1
```

Приклад вихідних даних::

```
29.0
```

Програма мовою C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   double x, y;
   cin >> x >> y;
   double c = x * x * x  + (x + 1) / (y * y + 1);
   cout << fixed << setprecision(1) << c;
}
```

Пояснення: піднесення до степення у цій програмі можна було використати функцію **pow()** із бібліотеки **<cmath>** . Тоді x^3 записали б так: **pow(x,3)**.

==========

Задача 7 (1006: Різниця чисел)

Дано цілі числа ***x , y***. Знайти різницю першого та другого числа.

Формат вхідних даних..

Стандартний потік містить цілі числа ***x , y*** **(0 <** ***x , y*** **<** 10^9**)** . Числа розділяються пропуском.

Формат вихідних даних.

Вивести одне число - різницю чисел

Приклад вхідних даних::

```
10 4
```

Приклад вихідних даних::

```
6
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x, y, c;
   cin >> x >> y;
   c = x - y;
   cout << c;
}
```

==========

Задача 8 (1007: Ціла частина)

Дано дійсне число ***N***. Вивести подвоєну його цілу частину.

Формат вхідних даних..

Стандартний потік містить ***N*** **(0 <** ***N*** **<** 10^9**)**

Формат вихідних даних.

Вивести подвоєну цілу частину ***N.***

Приклад вхідних даних::

```
12.958
```

Приклад вихідних даних::

```
24
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    double x;
    cin >> x;
    int c = 2 * (int)x;
    cout << c;
}
```

==========

Задача 9 (1008: Дробова частина)

Дано дійсні числа **P** і **K**. Дробову частину **P** помножити на **K** і вивести результат заокруглений до найближчого цілого.

Формат вхідних даних..

Стандартний потік містить числа **P** і **K (0 < P, K < 10^{12})**. Числа розділяються пропуском.

Формат вихідних даних.

Вивести одне ціле число

Приклад вхідних даних:

```
12.1257 1000
```

Приклад вихідних даних:

```
126
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    double p, k;
    cin >> p >> k;
    double c = p - (int)p; // отримали дробову частину
    c = c * k + 0.5; // округлення за правилами математики
    cout << (int)c; // вивід цілої частини
}
```

==========

Задача 10 (1009: Різниця площ).

Відомі сторони двох прямокутників a_1, b_1, a_2, b_2. Знайти модуль різниці їх площ.

Формат вхідних даних..

Стандартний потік містить цілі числа a_1, b_1, a_2, b_2. Сторони прямокутників не перевищують **1000**.

Формат вихідних даних.

Вивести одне число

Приклад вхідних даних::

```
2 3 1 8
```

Приклад вихідних даних::

```
2
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a1, b1, a2, b2;
   cin >> a1 >> b1 >> a2 >> b2;
   int  ans = abs(a1 * b1 - a2 * b2);
   cout << ans;
}
```

==========

Задача 11 (1010: Добуток цифр)

Для заданого двозначного числа знайти добуток його цифр.

Формат вхідних даних..

Вхідний потік містить двозначне число

Формат вихідних даних.

У вихідний потік вивести добуток цифр

Приклад вхідних даних::

```
34
```

Приклад вихідних даних::

```
12
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x;
   cin >> x;
   int a = x / 10; // кількість десятків
   int b = x % 10; // кількість одиниць
   int ans  =  a * b;
   cout << ans;
}
```

Пояснення: кількість рядків у коді цієї програми можна скоротити:

```
    int x;
    cin >> x;
    cout << (x / 10) * (x % 10);
```

==========

Задача 12 (1011: Обернений порядок цифр)

Формат вхідних даних..

Вхідний потік містить тризначне число

Формат вихідних даних.

Вивести через пропуск цифри в оберненому порядку

Приклад вхідних даних::

```
123
```

Приклад вихідних даних::

```
3 2 1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x, a, b, c;
   cin >> x;
   c = x % 10;
   x = x / 10;
   a = x / 10;
   b = x % 10;
```

```
    cout << c << ' ' << b << ' ' << a;
}
```

==========

Задача 13 (1012: Об'єм куба)

Знайти об'єм куба із стороною **a**.

Формат вхідних даних..

Стандартний потік містить натуральне число **a ≤ 100**

Формат вихідних даних.

Вивести об'єм куба.

Приклад вхідних даних::

```
2
```

Приклад вихідних даних::

```
8
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int a;
    cin >> a;
    cout << a * a * a;
}
```

==========

Задача 14 (1013: Середнє арифметичне)

Дано числа ***x, y***. Знайти їх середнє арифметичне.

Формат вхідних даних..

Стандартний потік містить цілі ***x, y*** по модулю не більші **1000**.

Формат вихідних даних.

Результат вивести з точністю до сотих.

Приклад вхідних даних::

```
2 3
```

Приклад вихідних даних::

```
2.50
```

Програма мовою С++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main(){
   int  a,b;
   cin >> a >> b;
   double s = (a + b)/ 2.0;
   cout << fixed << setprecision(2) << s;
}
```

==========

Задача 15 (1014: Переставити цифри)

Дано тризначне число. У ньому потрібно вилучили крайню ліву цифру і приписали її в кінець справа.

Формат вхідних даних.

Дано тризначне число

Формат вихідних даних.

Виведіть оновлене число. Якщо в утвореному числі спереду стане `0`, то його треба виводити.

Приклад вхідних даних::

```
243
```

Приклад вихідних даних::

```
432
```

Приклад вхідних даних::

```
105
```

Приклад вихідних даних::

```
051
```

Програма мовою С++:

```
#include <iostream>
using namespace std;
int main(){
   int n, a, b, c;
   cin >> n;
   c = n % 10;
   n = n / 10;
   b = n % 10;
```

```
    a = n / 10;
    cout << b << c << a;
}
```

Варіант другий:

```
#include <iostream>
using namespace std;
int main(){
    int n, a, b, c;
    cin >> n;
    b = n % 100;
    a = n / 100;
    cout << b / 10 << b % 10 << a;
}
```

Варіант 3: використання типу `string`:

```
#include <iostream>
using namespace std;
int main(){
    string  n;
    cin >> n;
    cout << n[1] << n[2] << n[0];
}
```

==========

Задача 16 (1015: Секунди в хвилини)

Від початку турніру з програмування пройшло ***N*** секунд. Виведіть кількість хвилин, які пройшли від початку турніру.

Формат вхідних даних.

Вхідний потік містить натуральне ***N*** **(*N* < 10^5).**

Формат вихідних даних.

Вивести кількість хвилин

Приклад вхідних даних:

150

Приклад вихідних даних::

2

Програма мовою C++:

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

```
int main(){
   int  n;
   cin >> n;
   cout << n / 60;
}
```

==========

Задача 17 (1016: Відрізки)

Дано цілі додатні числа ***A*** і ***B***. На відрізку довжиною ***A*** розмістили максимально можливу кількість відрізків довжиною ***B*** так, що вони не накладаються. Знайдіть довжину незайнятої частини відрізка ***A***.

Формат вхідних даних.

Вхідний потік містить цілі числа ***A, B*** $(0 < B < A < 10^6)$. Числа розділяються пропуском.

Формат вихідних даних.

Вивести ціле число - довжину незайнятої частини відрізка ***A***.

Приклад вхідних даних: :

```
5 2
```

Приклад вихідних даних::

```
1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main(){
   int  a,b;
   cin >> a >> b;
   cout << a % b;
}
```

==========

Програми на розгалуження

Задача 18 (1017: Парне-непарне)

Дано ціле число **X**.

Формат вхідних даних.

Вхідний потік містить ціле число **X (0 < X < 10^9)**

Формат вихідних даних.

Вивести **'Yes'**, якщо воно парне, або **'No'** коли **X** є непарним.

Приклад вхідних даних::

```
2
```

Приклад вихідних даних::

```
Yes
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int  x;
   cin >> x;
   if( x % 2 == 0) cout << "Yes\n";
     else cout << "No\n";

}
```

==========

Задача 19 (1018: Подвоїти більше)

Дано цілі числа ***a*** та **b**.

Формат вхідних даних.

Вхідний потік містить цілі числа ***a, b (a, b < 10^9)***. Числа розділяються пропуском.

Формат вихідних даних.

Вивести подвоєне більше число.

Приклад вхідних даних::

```
2 3
```

Приклад вихідних даних::

```
6
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int  a, b;
   cin >> a >> b;
   if( a > b) cout << 2*a <<'\n';
     else cout << 2*b <<'\n';

}
```

Варіант 2: Використання вбудованої функції max :

```
#include <iostream>
using namespace std;
int main()
{
   int  a, b;
   cin >> a >> b;
   cout << 2 * max(a, b) << '\n';
}
```

Варіант 3: використовуємо скорочений запис команди розгалуження.

```
#include <iostream>
using namespace std;
int main()
{
   int  a, b, c;
   cin >> a >> b;
   c = 2 * b;
   if( a > b) c = 2 * a;
   cout << c <<'\n';
}
```

==========

Задача 20 (1019: Частка)

Дано ціле числа **a**. Якщо воно парне - вивести його частку від ділення на **2**, інакше **0**.

Формат вхідних даних.

Вхідний потік містить **a (0 < a < 10^9)**

Формат вихідних даних.

Вивести одне число

Приклад вхідних даних:

```
10
```

Приклад вихідних даних:

```
5
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int  a, b = 0;
   cin >> a;
   if( a % 2 == 0) b = a / 2;
   cout << b << '\n';
}
```

==========

Задача 21 (1020: Впорядкування двох)

Дано цілі числа *a* та b. Вивести їх в один рядок в порядку зростання, тобто, спочатку менше, а потім більше.

Формат вхідних даних.

Вхідний потік містить цілі числа ***a, b*** **(0 < *a, b* < 10^9).** Числа розділяються пропуском.

Формат вихідних даних:

В єдиному рядку вивести два цілих числа розділених пропуском.

Приклад вхідних даних:

```
3 2
```

Приклад вихідних даних:

```
2 3
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int  a, b;
```

```
        cin >> a >> b;
        if( a > b) swap(a, b);
        cout << a << ' ' << b << '\n';
    }
```

==========

Задача 22 (1021: Чи ціле?)

Дано додатне число **P**. Вивести «**Yes**», якщо число ціле або «**No**» в іншому випадку.

Формат вхідних даних.

Вхідний потік містить **P (0 < P < 10^6)**

Формат вихідних даних:

Вивести **"Yes"** або **"No"**

Приклад вхідних даних:

```
12
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   double p;
   cin >> p;
   if( p == (int)p) cout << "Yes\n";
    else cout << "No\n";
}
```

==========

Задача 23 (1022: Найбільше та найменше)

Дано цілі числа ***a, b, c,*** **d**. В одному рядку вивести спочатку найменше з них, а потім найбільше.

Формат вхідних даних.

Вхідний потік містить чотири цілі числа ***a, b, c, d*** **(0 < *a, b, c, d* < 10^9)**, які розділяються пропуском.

Формат вихідних даних:

Вивести в єдиному рядку через пропуск найменше та найбільше з введених чисел

Приклад вхідних даних:

```
4 3 2 1
```

Приклад вихідних даних:

```
1 4
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c, d;
   cin >> a >> b >> c >> d;
   int mx = a; // нехай перше число є максимальним
   if ( b > mx) mx = b; // якщо b більше за mx, то воно
стає максимальним;
   if ( c > mx) mx = c;
   if ( d > mx) mx = d;
   int mi = a; // нехай перше число є мінімальним
   if ( b < mi) mi = b; // якщо b менше за mi, то воно
стає мінімальним;
   if ( c < mi) mi = c;
   if ( d < mi) mi = d;
   cout << mi << ' ' << mx << '\n';
}
```

Варіант 2:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c, d;
   cin >> a >> b >> c >> d;
   /*
     використовуємо вбудовані функції знаходження мінімального
     та максимального.
   */
   int mx = max(max(a, b), max(c, d));
   int mi = min(min(a, b), min(c, d));
   cout << mi << ' ' << mx << '\n';
}
```

==========

Задача 24 (1023: Координатна площина)

Задається точка з координатами ***x, y***. Точка не лежить на жодній з осей координат. Визначити, чи належить точка другій чверті.

Формат вхідних даних:

Вхідний потік містить два цілі числа ***x, y*** $(-10^9 < x, y < 10^9)$, які розділяються пропуском.

Формат вихідних даних:

Вивести «**Yes**», якщо точка лежить у другій чверті або «**No**» в іншому випадку.

Приклад вхідних даних:

```
-10 100
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x, y;
   cin >> x >> y;
   if ( x < 0 && y > 0) cout <<"Yes\n";
      else cout <<"No\n";
}
```

==========

Задача 25 (1024: Номер чверті)

Дано точка з координатами ***x, y***. Точка не лежить на жодній з осей координат. Вивести номер чверті, якій належить задана точка.

Формат вхідних даних.

Вхідний потік містить два цілі числа ***x, y*** $(-10^9 < x, y < 10^9)$, які розділяються пропуском.

Формат вихідних даних:

Вивести номер чверті, якій належить задана точка.

Приклад вхідних даних:

```
2 -8
```

Приклад вихідних даних:

4

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x, y, n = 4;
   cin >> x >> y;
   if ( x > 0 && y > 0) n = 1;
   if ( x < 0 && y > 0) n = 2;
   if ( x < 0 && y < 0) n = 3;
   cout << n <<"\n";

}
```

==========

Задача 26 (1025: Існування трикутника)

Дано цілі **x, y, z**. Чи існує трикутник з такими сторонами?

Формат вхідних даних.

Вхідний потік містить цілі числа **x, y, z (0 < x, y, z < 10^9)**, які розділяються пропуском.

Формат вихідних даних:

Вивести **"Yes"** або **"No"** - відповідь на поставлене запитання..

Приклад вхідних даних:

1 4 5

Приклад вихідних даних:

No

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c;
   cin >> a >> b >> c;
/*
Виконаємо неповне сортування трійки чисел a, b, c, так щоб
змінна c отримала максимальне значення.
```

```
*/
   if ( a > b) swap(a, b);
   if ( b > c) swap(b, c);
// Перевірка умови трикутника
   if ( a + b > c) cout << "Yes\n";
   else cout << "No\n";
}
```

Варіант 2:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c;
   cin >> a >> b >> c;
// Перебір усіх варіантів для умови трикутника
   if ( a + b > c && a + c > b && c + b > a )
        cout << "Yes\n";
   else cout << "No\n";
}
```

==========

Задача 27 (1026: Чи зростаюча?)

Дано цілі числа ***a, b, c, d***. Чи утворюють ці числа, у такому слідуванні, зростаючу послідовність?

Формат вхідних даних.

Вхідний потік містить чотири цілі числа **a, b, c, *d*** **(-10^9 < a, 6, c, *d* < 10^9)**, які розділяються пропуском.

Формат вихідних даних:

Вивести «**Yes**» або «**No**» - відповідь на поставлене запитання.

Приклад вхідних даних:

```
3 4 4 6
```

Приклад вихідних даних:

```
No
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    if(a < b && a < c && a < d && b < c && b<d && c<d)
     cout << "Yes\n";
     else cout << "No\n";
}
```

==========

Задача 28 (1027: Найбільша цифра)

Задано натуральне число ***N***. Визначити найбільшу цифру цього числа.

Формат вхідних даних.

Вхідний потік містить натуральне число ***N*** **(1000 < *N* < 10000)**

Формат вихідних даних:

Вивести найбільшу цифру

Приклад вхідних даних:

1234

Приклад вихідних даних:

4

Програма мовою C++:

Використовуємо стандартну функцію **max()**, яка знаходить максимальне із двох чисел:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c, d, n;
   cin >> n;
   d = n % 10;
   n = n / 10;
   c = n % 10;
   n = n / 10;
   b = n % 10;
   a = n / 10;
   n = max(max(a,b),max(c,d));
   cout << n << "\n";
}
```

Варіант 2: Сортуємо знайдені цифри:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c, d, n;
   cin >> n;
   d = n % 10;
   n = n / 10;
   c = n % 10;
   n = n / 10;
   b = n % 10;
   a = n / 10;
   if ( a > b) swap(a,b);
   if ( b > c) swap(c,b);
   if ( c > d) swap(c,d);
   cout << d << "\n";
}
```

==========

Задача 29 (1028: Сума чи різниця)

Задано двозначне натуральне число ***N***. Якщо квадрат цього числа дорівнює кубу суми його цифр, то вивести суму цифр числа, а якщо ні, то вивести модуль різниці цифр.

Формат вхідних даних.

Вхідний потік містить натуральне ***N*** **(10 < *N* < 100).**

Формат вихідних даних:

Вивести одне число - відповідь на поставлену задачу

Приклад вхідних даних:

```
11
```

Приклад вихідних даних:

```
0
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, n;
   cin >> n;
   // знайдемо цифри двозначного числа
```

```
        b = n % 10;
        a = n / 10;
        int s = a + b;
        s = s * s * s; // куб суми
        int res = abs(a - b); // відповідь у випадку не
    виконання умови
        if ( n * n == s ) res = a + b;
        cout << res << "\n";
    }
```

==========

Задача 30 (1029: Значення виразу)

Обчислити значення виразу: $y=\frac{x^3+2}{x^2-1}+5$

Результат вивести з трьома знаками після коми. Якщо вираз обчислити неможливо, то вивести **"No"**.

Формат вхідних даних.

Вхідний потік містить **x (-10⁶ < x < 10⁶)**

Формат вихідних даних:

Вивести значення виразу або **"No"**.

Приклад вхідних даних:

```
2
```

Приклад вихідних даних:

```
8.333
```

Програма мовою C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   double x, y;
   cin >> x ;
   if ( fabs(x) == 1) cout << "No\n";
   else
   {
       y = (x * x *x + 2) / (x * x - 1) + 5;
       cout << fixed << setprecision(3) << y << '\n';
```

```
    }
}
```

==========

Задача 31 (1030: Ділення цифр)

Дано ціле число ***N***. Знайти неповну частку від ділення першої цифри на другу.

Якщо це зробити неможливо, то вивести "**NO**".

Формат вхідних даних.

Вхідний потік містить ціле число ***N*** **(10 < *N* < 100)**

Формат вихідних даних:

Вивести шукану частку або "**NO**"

Приклад вхідних даних:

84

Приклад вихідних даних:

2

Програма мовою C++:

```
    #include <iostream>
    using namespace std;
    int main()
    {
       int n, a, b;
       cin >> n ;
       b = n % 10;
       a = n / 10;
       if ( b == 0) cout << "NO\n";
       else  cout <<  a / b << '\n';
    }
```

Варіант 2:

```
    #include <iostream>
    using namespace std;
    int main()
    {
       string  n; //число прочитали як рядок
       cin >> n;
       int b = n[1] - '0';
       int a = n[0] - '0';
       if ( b == 0) cout << "NO\n";
       else  cout <<  a / b << '\n';
    }
```

==========

Задача 32 (1031: Шахи: тура)

Дано координати двох клітинок шахової дошки. Координати задаються числами. Вивести «**Yes**», якщо тура може за один хід перейти з однієї клітинки в іншу. В іншому випадку вивести «**No**».

Формат вхідних даних.

Вхідний потік містить чотири цілі числа від **1** до **8** - координати першої та другої клітинки. Числа розділяються пропуском.

Формат вихідних даних:

Вивести "**Yes**" або "**No**" - відповідь на завдання.

Приклад вхідних даних:

```
3 1 1 2
```

Приклад вихідних даних:

```
No
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int x1,y1,x2,y2;
   cin >> x1 >> y1 >> x2 >> y2;
   if ( x1 == x2 || y1 == y2) cout << "Yes\n";
   else  cout << "No\n";
}
```

==========

Задача 33 (1032: Крайні точки)

На осі *Ox* вказані три різні точки: X_1, X_2, X_3. Вивести значення тих двох точок, між якими лежить третя. Координати точок є цілими числами.

Формат вхідних даних.

Вхідний потік містить три цілі числа $X_1, X_2, X_3 (-10^9 < X_1, X_2, X_3 < 10^9)$, які розділяються пропуском.

Формат вихідних даних:

В одному рядку через пропуск вивести координати шуканих точок у порядку зростання координати.

Приклад вхідних даних:

```
2 5 1
```

Приклад вихідних даних:

```
1 5
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c;
   cin >> a >> b >> c;
   if ( a > b) swap(a, b);
   if ( b > c) swap(c, b);
   if ( a > b) swap(a, b);
   cout << a << ' ' << c << '\n';
}
```

==========

Задача 34 (1033: Діляться на 5?)

Задаються числа ***a, b, c, d*** із яких хоча б одне закінчується нулем. Вивести числа, які кратні **5**.

Формат вхідних даних.

Вхідний потік містить цілі числа ***a,b,c,d*** **(0 < *a,b,c,d* < 10^9)**, які розділяються пропуском.

Формат вихідних даних:

Вивести числа кратні **5** у порядку їх слідування.

Приклад вхідних даних:

```
5 30 2 15
```

Приклад вихідних даних:

```
5 30 15
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
```

```
{
   int a;
   cin >> a;
   if ( a % 5 == 0) cout << a << ' ';
   cin >> a;
   if ( a % 5 == 0) cout << a << ' ';
   cin >> a;
   if ( a % 5 == 0) cout << a << ' ';
   cin >> a;
   if ( a % 5 == 0) cout << a << ' ';
}
```

Примітка: до цієї задачі слід повернутися при вивченні циклів. ЇЇ розв'язок може бути таким:

```
#include <iostream>
using namespace std;
int main()
{
   int a, t = 4;
   while (t--)
    {
        cin >> a;
        if ( a % 5 == 0) cout << a << ' ';
    }
}
```

Або таким:

```
#include <iostream>
using namespace std;
int main()
{
   int a;
   while (cin >> a)
     if ( a % 5 == 0) cout << a << ' ';
}
```

При перевірці, програму слід перервати комбінацією клавіш: **Ctrl+Z**

==========

Задача 35 (1034: Який трикутник?)

Дано три відрізки із довжинами ***x, y, z.***

Вивести:

`0`, якщо ці відрізки не можуть утворити трикутник;

`-1`, якщо утворюють довільний трикутник;

`1`, якщо утворюють прямокутний трикутник.

Формат вхідних даних.

Вхідний потік містить три цілі числа x, y, z ($0 < x, y, z < 10^6$), які розділяються пропуском.

Формат вихідних даних:

Вивести 0, або 1, або -1 - відповідь на поставлене завдання.

Приклад вхідних даних:

```
3 4 5
```

Приклад вихідних даних:

```
1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int a, b, c;
   cin >> a >> b >> c;
// виконаємо часткове сортування трьох чисел
   if( a > b) swap(a, b);
   if( b > c) swap(c, b);
   int res = 0;
   if( a + b > c)
   {
       res = -1;
       if( a * a + b * b == c * c) res = 1;
   }
   cout << res << '\n';
}
```

==========

Задача 36 (1037: Прямокутник)

Петрику потрібно вибрати на площині `4` точки так, щоб вони утворювали прямокутник зі сторонами, паралельними осям координат. Петрик вже вибрав

три точки і впевнений, що він вибрав їх вірно. Допоможіть Петрику знайти координати четвертої точки.

Формат вхідних даних.

Вхідний потік містить три рядки. Кожен рядок містить два натуральних числа, відокремлених пропуском - координати однієї з вершин прямокутника. Всі координати лежать у діапазоні від **1** до **1000**.

Формат вихідних даних:

Вивести два цілих числа - координати четвертої вершини прямокутника.

Приклад вхідних даних:

```
5 5
5 7
7 5
```

Приклад вихідних даних:

```
7 7
```

Програма мовою C++:

Ідея розв'язку ґрунтується на переборі усіх варіантів розташування вершин прямокутника.

```
#include <iostream>
using namespace std;
int main()
{
   int x1, y1, x2, y2, x3, y3, x = 0, y = 0;
   cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
   if (x1 == x2) x = x3;
   if (x1 == x3) x = x2;
   if (x3 == x2) x = x1;
   if (y1 == y2) y = y3;
   if (y1 == y3) y = y2;
   if (y3 == y2) y = y1;
   cout << x << ' ' << y << '\n';
}
```

Якщо врахувати те, що сторони прямокутника паралельні осям координат, то має бути дві пари однакових x та дві пари однакових y. Згадаємо бітову операцію xor

(у **C++** це ^) і її основну властивість: **a ^ a = 0**, то розв'язок стає набагато простішим і переходить у категорію лінійних програм.

```
#include <iostream>
using namespace std;
int main()
{
   int x1, y1, x2, y2, x3, y3, x, y;
   cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
   x = x1 ^ x2 ^ x3;
   y = y1 ^ y2 ^ y3;
   cout << x << ' ' << y << '\n';
}
```

Цикли

Задача 37 (1035: Хлібовоз)

Степан працює на хлібовозі. Сьогодні у нього є рівно n буханок хліба. У першому магазині Степан вивантажує половину буханок плюс половину буханки (Степан любить залишати у магазинах не ціле число буханок). У наступному магазині він знову залишає половину буханок плюс одну половину. Так Степан поступив у кожному з k магазинів і у нього в хлібовозі не залишилося жодної буханки хліба.

Знайдіть скільки буханок було у хлібовозі на початку поїздки.

Формат вхідних даних.

Перший рядок містить кількість тестів t. Кожен тест містить в окремому рядку кількість зупинок k ($1 < k < 30$).

Формат вихідних даних:

Для кожного тесту вивести в окремому рядку початкову кількість буханок.

Приклад вхідних даних:

```
2
1
3
```

Приклад вихідних даних:

```
1
7
```

Програма мовою C++:

Ідея розв'язку:

Зрозуміло, що відповідь має бути числом непарним. При кожному діленні навпіл отримуємо також число непарне (бо віддаємо щоразу півбуханки) – це означає, що відповідь це таке число, двійковий запис якого, містить лише одиниці. Оскільки маємо k магазинів, то кількість буханок: $n = 2^k - 1$;

Задача на мультитест.

```cpp
#include <iostream>
using namespace std;
int main()
{
    int t; cin >> t;
    while(t--)
    {
        int k; cin >> k;
        int n = 1 << k; // побітовий зсув вліво
        cout << n - 1 << '\n';
    }
}
```

==========

Задача 38 (1036: Втрачена картка)

Для настільної гри використовуються картки з номерами від **1** до ***n*.** Одна картка загубилась. Знайдіть її.

Формат вхідних даних.

Вхідний потік містить в одному рядку числа: першим записане число ***n*** **(0 <** ***n*** **< 10^6)**, далі йдуть ***n*** **−** 1 номерів наявних карток.

Формат вихідних даних:

Вивести номер загубленої картки.

Приклад вхідних даних:

```
5 1 2 3 4
```

Приклад вихідних даних:

```
5
```

Програма мовою C++:

<u>Ідея розв'язку:</u>

Знайдемо суму всіх натуральних чисел від **1** до **n** за формулою:

```cpp
s = (1 + n) * n / 2;
```

А далі від цієї суми віднімаємо номери карток, що залишились.

```cpp
#include <iostream>
using namespace std;
int main()
{
  int n; cin >> n;
  long long s = (1LL + n) * n / 2;
```

```
    n--;
    while( n-- )
    {
        int x; cin >> x;
        s = s - x;
    }
    cout << s;
}
```

Варіант 2:

Використаємо операцію xor. Проксоримо усі числа від **1** до **n**. А далі зчитуємо номер картки і ксоримо її із отриманим ксором. Зрозуміло, що усі номери карток, які є у наявності, обнуляться (за властивістю: **a xor a = 0**) і залишиться номер втраченої картки.

```
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    int s = n, i = 1;
    while(i < n)
    {
        int x; cin >> x;
        s = s ^ x ^ i;
        i++;
    }
    cout << s;
}
```

==========

Задача 39 (1038: Кубики)

Дома у Ілька було **2** однакових набори кубиків з англійських літер, але під час чергового прибирання один з кубиків загубився. Допоможіть Ільку визначити, який саме з кубиків відсутній у одному з наборів.

Формат вхідних даних.

У першому рядку задано кількість знайдених Ільком кубиків **n (1 < n < 10^5)**, а у другому рядку - символи, зображені на кожному із знайдених кубиків.

Формат вихідних даних:

Виведіть літеру, зображену на загубленому кубику, або повідомлення "**Ok**", якщо Ілько помилився і жоден з кубиків не загубився.

Приклад вхідних даних:

5

abcac

Приклад вихідних даних:

b

Програма мовою С++:

Ідея розв'язку:

Кожна літера у рядку повинна мати пару, та літера, яка не має пари і є тим кубиком, який загубився. Оскільки, кожна літера має свій числовий код, то літери можна ксорити так як числа.

```
#include <iostream>
using namespace std;
int main()
{
  int n;  string s;
  cin >> n >> s;
  int k = 0;
  for ( char c: s) k = k ^ c;
  if (k)cout << (char)k;
   else cout << "Ok\n";
}
```

Примітка: при такій реалізації змінна n не використовується.

==========

Задача 40 (1039: Одиниці)

Задано ціле число **X** записане в десятковій системі числення. Напишіть програму, яка знаходить кількість одиниць, в його двійковому записі.

Формат вхідних даних.

У вхідному потоці міститься $X\ (0 < X < 10^{20})$.

Формат вихідних даних:

У стандартний вихідний потік вивести кількість одиниць у двійковому записі числа.

Приклад вхідних даних:

```
7
```

Приклад вихідних даних:

```
3
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  long long n;
  cin >> n;
  int k = 0;
  while ( n != 0)
  {
      k += n % 2;
      n = n / 2;
  }
  cout << k;

}
```

Та ж сама ідея, але з використанням бітових операцій:

```
#include <iostream>
using namespace std;
int main()
{
  long long n;
  cin >> n;
  int k = 0;
  while ( n )
  {
      k += n & 1;
      n >>= 1;
  }
  cout << k;

}
```

Примітка: виграш у часі отримали в `1/1000 c.`

==========

Задача 41 (1040: Max-Min)

Вивчаючи двійкову систему числення. Василько вирішив попрактикуватися, і придумав таку вправу. Він із бітів числа створював найбільше і найменше число, переставляючи біти, після чого знаходив їх різницю. Проте хлопець не знає, чи правильно виконує вправу. Допоможіть йому. Напишіть програму, яка за даним числом **N** знаходить різницю між найбільшим і найменшим числом, які утворюються із бітів заданого числа.

Пояснення. $N = 13_{10}$, в двійковій системі числення - 1101_2 - найбільше число $1110_2 = 14_{10}$ - найменше число $0111_2 = 7_{10}$. $14-7=7$.

Формат вхідних даних.

Вхідний потік містить ціле число N $(0 < N < 2^{31})$.

Формат вихідних даних:

Вивести одне число - відповідь до вправи Василька.

Приклад вхідних даних:

```
13
```

Приклад вихідних даних:

```
7
```

Програма мовою C++:

Ідея розв'язку: Нехай у числі x маємо **k** одиниць, а всього двійкових розрядів **n**. Наприклад, при **x = 358, k = 5**, а **n = 9**. Зрозуміло, що найменше число, це таке число у якому усі одиниці будуть справа, а найбільше – усі одиниці зліва. Якщо одиницю зсунути на **k** позицій вліво, то отримаємо 2^k. Мінімальне число буде рівне: **mi = 2^k – 1**. Максимальне число отримаємо, якщо мінімальне зсунемо вліво на **n - k** позицій.

```
#include <iostream>
using namespace std;
int main()
{
  int x;
  cin >> x;
  int k = 0, n = 0;
```

```
  while ( x )
  {
       n++; // кількість двійкових розрядів
       k += x & 1; // кількість одиниць у двійковому записі
       x >>= 1; // ділимо число на 2
  }
  int mi = ( 1 << k ) - 1; // знаходимо мінімальне
  int mx = mi << ( n - k );// знаходимо максимальне
  cout << mx - mi;

}
```

==========

Задача 42 (1041: Циклічні зсуви)

Натуральне число ***n*** переведемо у двійкову систему числення і утворимо всі ліві циклічні зсуви числа ***n***, при яких перша цифра числа переноситься в кінець числа. Наприклад, якщо ***n*** **= 11**. в двійковій системі буде **1011**, його циклічні зсуви: **0111, 1110, 1101, 1011**. Максимальне значення ***m*** з усіх отриманих у такий спосіб чисел буде мати число $\mathbf{1110_2 = 14_{10}}$. Для заданого числа ***n*** визначити максимальне значення ***m***.

Формат вхідних даних.

Вхідний потік містить ціле число $\boldsymbol{n}$ **(1 <** $\boldsymbol{n}$ **<** $\mathbf{2 \cdot 10^9}$**).**

Формат вихідних даних:

Вивести шукане число ***m***.

Приклад вхідних даних:

11

Приклад вихідних даних:

14

Програма мовою C++:

Ідея розв'язку: Не порушуючи загальності будемо моделювати процес зсуву на прикладі одного байту. У пам'яті комп'ютера десяткове число $\mathbf{x = 11_{10}}$ в межах одного байту матиме вигляд; **00001011**. Оскільки діючих розрядів в числі чотири, **k = 4,** то необхідно побудувати число **h,** яке матиме вигляд: **00001111.** Число будуємо так: **h = (1 << k) – 1.** При побітовому множенні числа **x** на **h** будемо отримувати нове число, але в межах чотирьох

розрядів. При циклічному зсуві, потрібно знати, який **bit**, підлягає переносу: одиниця чи нуль. Це взнаємо, коли число виду $2^k = h + 1$ побітово помножимо на число **x << 1**, яке зсунули вправо на **1**. На початку відповідь, тобто максимальне число, ініціалізуємо заданим числом. Зрозуміло, що для знаходження відповіді **mx** , досить зробити **k** циклічних зсувів.

```
#include <iostream>
using namespace std;
int main()
{
     int n; cin >> n;
     int k = 0, mx = n, x = n;
     // взнаємо кількість розрядів у числі n
     while ( n != 0)
     {
          k++;
          n = n >> 1;
     }
     int h = 1 << k, bit = 0; //Знаходимо h = 2^k
     for(int i = 0; i < k; i++)//виконуємо k циклічних зсувів
     {
          x = x << 1; // циклічний зсув числа
          bit = 0;
          if( x & h) bit = 1; // знайдемо значення bit
          //побітове множення на h - 1 та добавляємо bit
          x = ((h - 1) & x) + bit;
          mx = max(mx,x); // перераховуємо відповідь
     }
     cout << mx;
}
```

==========

Задача 43 (1042: Доповнюваний код-1)

Напишіть програму, яка за заданими числами ***A*** та ***n*** записує подання числа ***A*** у ***n*** - розрядному двійковому доповнюваному коді.

Формат вхідних даних.

Перший рядок вхідних даних містить число ***A*** - другий рядок - число ***n***, при цьому $2 < n < 16$, $-2^{n-1} < A < 2^{n-1} - 1$.

Формат вихідних даних:

Програма повинна вивести рядок з ***n*** символів, який містить запис числа ***A*** у ***n*** - розрядному двійковому доповнюваному коді, перший символ - старший знаковий розряд.

Приклад вхідних даних:

```
5
8
```

Приклад вихідних даних:

```
00000101
```

Приклад вхідних даних:

```
-5
8
```

Приклад вихідних даних:

```
11111011
```

Програма мовою C++:

Ідея розв'язку: Врахуємо те, що в пам'яті компютера число уже зберігається у двійкому коді, а тому досить **n** разів повторити наступні дії:

- взнати який одиничний біт числа;
- знайдений біт перетворити у символ;
- утворений символ конкатенувати до рядка зліва;
- зсунути вправо на один розряд задане число.

```
#include <iostream>
using namespace std;
int main()
{
    int x, n; cin >> x >> n;
    string s = "";// на початку рядок порожній
    while( n-- )// повторюємо n разів
    {
    int b = x & 1; // взнали біт
    char ch = (char)(b + 48);// перетворили у символ
    s = ch + s;//додали до рядка
    x = x >> 1; // зсув в право
    }
```

```
    cout << s;
}
```

==========

Задача 44 (1043: Доповнюваний код-2)

Дано запис деякого числа у двійковому доповнюваному коді. Виведіть десятковий запис цього числа.

Формат вхідних даних.

Вхідний потік містить рядок `S (2 < |S| < 16)`, який складається з `0` та `1`.

Формат вихідних даних:

Виведіть число в десятковому запису.

Приклад вхідних даних:

```
00000101
```

Приклад вихідних даних:

```
5
```

Приклад вхідних даних:

```
11111011
```

Приклад вихідних даних:

```
-5
```

Програма мовою C++:

Ідея розв'язку: Якщо старший розряд нуль, то число додатнє, в противному випадку – число із знаком мінус. Для від'ємного числа виконаємо реверс двійкового коду. За допомогою функції переведемо двійковий рядок у десяткове число.

```
#include <iostream>
using namespace std;
int covert( string s)
{
    int d = 0,c = 1;
    int k = s.size();
    for( int i = k - 1; i >= 0; i--)
    {
        int a = s[i] - 48;
        d = d + a * c;
        c = c << 1;
```

```
    }
    return d;
}
int main()
{
    string s; cin >> s;
    int n = s.size();
    int ans = covert(s);
    if ( s[0] == '1')
    {
        for(int i = 0; i < n; i++)
            if (s[i] == '0') s[i] = '1';
               else s[i] = '0';
        ans = -(covert(s) + 1) ;
    }
    cout << ans <<'\n';
}
```

==========

Задача 45 (1044: Одиниці та сімки)

Знайти **N**-й член зростаючої послідовності натуральних чисел: **1 7 11 1771 77 111 117 171 177 711 717..**

Формат вхідних даних.

Вхідний потік містить натуральне число **N (1 < N < 10^7).**

Формат вихідних даних:

Вивести відповідь до поставленої задачі

Приклад вхідних даних:

7

Приклад вихідних даних:

111

Приклад вхідних даних:

12

Приклад вихідних даних:

717

Програма мовою C++:

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

```
int main()
{
    int n; cin >> n;
    n++;
    string s = "";
    while( n != 1)
    {
       if ( n & 1) s = '7' + s;
        else s = '1' + s;
        n = n >> 1;
    }
    cout << s << '\n';
}
```

==========

Задача 46 (1045: Мутанти)

Генетичний код кролика являє собою деяке двійкове число. Небезпечний вірус уражає кролика змінюючи його код. Вірус змінює справа (необов'язково з кінця) одиничні біти на нульові, так щоб утворився неперервний ланцюжок нульових бітів. У фермера Вампірова є **N** уражених кроликів, допоможіть йому взнати нові генетичні коди кроликів, для кращого та найшвидшого їх лікування. Якщо новий генетичний код кролика **0** - то такий кролик є невиліковним.

Пояснення:

Нехай код ураженого кролика $86_{10} = 1010110_2$. *Новий код* $80_{10} = 1010000_2$. *Кролик з кодом* $56_{10} = 111000_2$ - *є невиліковним*.

Формат вхідних даних.

Перший рядок стандартного потоку містить натуральне число **N(N < 10^3)** - кількість кроликів. У наступному рядку через пропуск записано **N** натуральних чисел, кожне із яких не перевищує **$2 \cdot 10^9$**

Формат вихідних даних:

У перший рядок вихідного стандартного потоку записати через пропуск нові ненульові генетичні коди кроликів, а у другий рядок - кількість невиліковних кроликів.

Приклад вхідних даних:

4

```
23 56 36 86
```

Приклад вихідних даних:

```
16 32 80
1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int f ( int x)
{
    int k = 0;
// поки число парне будемо його ділити на 2 і підраховувати кількість поділів
    while( !(x & 1))
    {
        k++;
        x = x >> 1;
    }
    int c = 0;
// поки число не парне будемо його ділити на 2 і підраховувати кількість поділів

    while ( x & 1)
    {
        c++;
        x = x >> 1;
    }
// результатом буде число x помножено на 2^(k+c).
    return (x <<( k + c));
}
int main()
{
  int n, cnt = 0; cin >> n;
  for (int i = 0; i < n; i++)
  {
      int a; cin >> a;
      int b = f(a);
      if( b ) cout << b <<' ';
        else cnt++;
  }
  cout << '\n' << cnt << '\n';
}
```

==========

Задача 47 (1046: Кількість парних)

Дано ***N*** цілих чисел. Знайти кількість парних серед них.

Формат вхідних даних.

У першому рядку стандартного вхідного потоку міститься ***N*** **(1< *N* <1000)** - кількість чисел. У наступному рядку через пропуск дано цілі додатні числа не більші **10000**, які розділяються пропуском.

Формат вихідних даних:

У стандартний вихідний потік вивести кількість парних чисел.

Приклад вхідних даних:

```
5
12 4 2 7 9
```

Приклад вихідних даних:

```
3
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int n;  cin >> n;
  int k = 0;
  while ( n-- )
  {
      int x; cin >> x;
      k += 1 - (x & 1);
  }
  cout << k << '\n';
}
```

Примітка: вираз: **x & 1** - можна замінити на: **x % 2**.

Варіант із використанням циклу **for**:

```
#include <iostream>
using namespace std;
int main()
{
  int n;  cin >> n;
  int k = 0;
```

```
    for (int i = 0; i < n; i++)
    {
        int x; cin >> x;
        k += (x & 1);
    }
    cout << n - k << '\n';
}
```

==========

Задача 48 (1047: Сума трицифрових)

Дано ***N*** цілих чисел. Знайти суму трицифрових чисел, що є серед даного набору.

Формат вхідних даних.

У першому рядку стандартного вхідного потоку міститься ціле ***N*** (1 < ***N*** < 10^3) - кількість чисел. У наступному рядку через пропуск дано цілі додатні числа не більші 10^4, які розділяються пропуском.

Формат вихідних даних:

У стандартний вихідний потік вивести суму трицифрових чисел.

Приклад вхідних даних:

```
5
12 4 200 7 101
```

Приклад вихідних даних:

```
301
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n, s = 0;
    cin >> n;
    for (int i = 0; i < n; i++)
    {
        int x; cin >> x;
        if (x > 99 && x < 1000) s = s + x;
    }
    cout << s << '\n';
}
```

==========

Задача 49 (1048: Непарні до квадрату)

Дано ***N*** цілих чисел. Всі непарні числа підняти до квадрату.

Формат вхідних даних.

У першому рядку стандартного вхідного потоку міститься ціле ***N*** **(1 < *N* < 1000)** - кількість чисел. У наступному рядку через пропуск дано цілі додатні числа не більші **10000**, які розділяються пропуском.

Формат вихідних даних:

У стандартний вихідний потік вивести через пропуск квадрати непарних чисел.

Приклад вхідних даних:

```
5
12 4 2 7 9
```

Приклад вихідних даних:

```
49 81
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int n;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
      int x; cin >> x;
      if (x & 1) cout << x * x << ' ';
  }
}
```

==========

Задача 50 (1049: Кратні 5)

Дано ***N*** цілих чисел. Знайти суму чисел, які кратні **5**.

Формат вхідних даних.

У першому рядку стандартного вхідного потоку міститься ціле ***N*** **(1 < *N* < 1000)** - кількість чисел. У наступному рядку через пропуск дано цілі додатні числа не більші **10000**, які розділяються пропуском.

Формат вихідних даних:

У стандартний вихідний потік вивести суму чисел.

Приклад вхідних даних:

```
5
15 4 2 10 9
```

Приклад вихідних даних:

```
25
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int n, s = 0;
  cin >> n;
  for (int i = 0; i < n; i++)
  {
      int x; cin >> x;
      if (x % 5 == 0) s = s + x;
  }
  cout << s << '\n';
}
```

==========

Задача 51 (1053: Прості дільники)

Знайти прості дільники числа ***N***.

Формат вхідних даних.

У стандартному вхідному потоці дано ціле додатне N $(1 < N < 10^9)$.

Формат вихідних даних:

У стандартний вихідний потік вивести через пропуск його прості дільники у неспадному порядку. Якщо число ***N*** ділиться на деяке просте число більше одного разу, то виводити цей дільник також більше одного разу.

Приклад вхідних даних:

```
20
```

Приклад вихідних даних:

```
2 2 5
```

Програма мовою C++:

Ідея розв'язку: усі прості дільники числа **n**, не перевищують кореня квадратного із **n.**

```
#include <iostream>
using namespace std;
int main()
{
    long long n;  cin >> n;
    for ( int i = 2; i * i <= n; i++)
        while ( n % i == 0)
         {
            n = n / i;
            cout << i <<' ';
         }
    if ( n != 1) cout << n;
}
```

==========

Задача 52 (1066: Шнурки)

Свої елітні черевики Дмитрик зашнуровує елітним способом: вгору, по горизонталі в інший ряд. вгору, по горизонталі і так далі. Якою повинна бути довжина шнурка для його черевик, якщо відомо, що дірочки для шнурування розміщенні у два ряди, відстань між рядами дорівнює ***A***, а відстань між дірочками у ряді ***B***. кількість дірочок у кожному ряді ***N***. Крім того Діма любить зав'язати шнурок елітним бантиком, а для цього необхідно щоб довжина вільного кінця була ***L***?

Формат вхідних даних.

Зі стандартного вхідного потоку прочитати чотири натуральних числа ***A, B, L, N***. Всі числа не перевищують 10^9.

Формат вихідних даних:

У стандартний вихідний потік вивести відповідь на задачу.

Приклад вхідних даних:

```
2 1 3 4
```

Приклад вихідних даних:

```
26
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    long long a,b,L,n,res;
    cin >> a >> b >> L >> n;
    res =2* ((n - 1)*(a +b) + L) + a;
    cout << res << endl;
}
```

==========

Задача 53 (1069: Цифри у зворотному порядку)

Дано натуральне число ***N***.

Формат вхідних даних.

Вхідний потік містить натуральне число N $(1 \le N \le 2 \cdot 10^9)$

Формат вихідних даних:

Вивести його цифри через пропуск у зворотному порядку.

Приклад вхідних даних:

```
1239
```

Приклад вихідних даних:

```
9 3 2 1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n;   cin >> n;
    while ( n )
    {
        cout << n % 10 << ' ';
        n = n / 10;
    }
}
```

==========

Задача 54 (1071: Степінь 2)

Дано натуральне число N, що є деякою степеню числа 2: $N=2^k$. Знайдіть ціле число **k**.

Формат вхідних даних.

У стандартному потоці міститься N $(1 \le N \le 2 \cdot 10^9)$.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

```
8
```

Приклад вихідних даних:

```
3
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n, k = 0;
    cin >> n;
    while ( n != 1 )
    {
        k++;
        n = n / 2;
    }
    cout << k <<'\n';
}
```

==========

Задача 55 (1072: Сума послідовності-2)

Дано натуральне число N. Вивести в одному рядку найменше з цілих k. для якого сума $1 + 2 + \ldots + k$ буде більша або рівна N і саму цю суму.

Формат вхідних даних.

У стандартному потоці міститься N $(1 \le N \le 2 \cdot 10^9)$.

Формат вихідних даних:

У стандартний потік вивести результат: два числа в одному рядку через пропуск.

Приклад вхідних даних:

```
10
```

Приклад вихідних даних:

```
4 10
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int n, k = 0, s = 0;
  cin >> n;
  while (  s < n )
    {
        k++;
        s += k;

    }
   cout <<  k << ' ' << s << '\n';
}
```

==========

Задача 56 (1077: Кількість чисел послідовності)

Задається N цілих додатних чисел не більших 1000. Серед них знайти кількість чисел, що є членами такої послідовності $k^2 + k + 1$, де $k = 0, 1, 2$ …

Формат вхідних даних.

У першому рядку міститься ціле число N $(1 \le N \le 1000)$. У наступному рядку містяться цілі додатні числа не більші 10^4, які розділяються пропуском.

Формат вихідних даних:

У вихідний потік вивести кількість чисел, що відповідають умові задачі.

Приклад вхідних даних:

5

3 5 6 7 8

Приклад вихідних даних:

2

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int n, cnt = 0;
  cin >> n;
  while (  n-- ) // пока є числа у потоці
```

```
        {
            int x; cin  >> x; // читаємо чергове число x
            int k = 0;
            while ( k*k+k+1 < x) k++; // знаходимо k
            // якщо k=x то збільшуємо лічильник
             if( k*k+k+1 == x) cnt++;
        }
       cout <<  cnt << '\n'; // виводимо відповідь
    }
```

==========

Задача 57 (1079: Найбільша сума цифр)

Серед ***N*** цілих додатних чисел не більших **1000** знайти числа з найбільшою сумою цифр та найменшою. Якщо таких чисел є декілька, то слід вибирати ті, що ідуть у переліку першими.

Формат вхідних даних.

У першому рядку задано ціле число ***N*** **(1 < *N* < 1000)**. У наступному рядку містяться цілі додатні числа не більші **10000**, які розділяються пропуском.

Формат вихідних даних:

У вихідний потік вивести два числа через пропуск, що відповідають умові задачі. Спочатку вивести число з найбільшою сумою цифр, а потім з найменшою. Якщо є два числа, що задовольняють одну із умов, то слід виводити те, що іде першим у порядку слідування.

Приклад вхідних даних:

```
5
12 10 101 1000 102
```

Приклад вихідних даних:

```
12 10
```

Програма мовою C++:

Ідея розв'язку: для кожного числа з потоку знаходимо суму його цифр і підтримуємо максимум та мінімум із усіх сум. На початку максимум **mx = 0**, а мінімум **mi > 45** (бо найбільше число п'ятизначне, а його найбільша можлива сума цифр **45**).

```
#include <iostream>
using namespace std;
int main()
{
  int n, mx = 0, mi = 50,x_max, x_min;
  cin >> n;
  while (  n-- )
    {

        int x; cin  >> x;
        int s = 0;
        int X = x; // копія поточного числа
        // знаходимо суму цифр числа x:
         while ( x )
             {
                 s += x % 10;
                 x /= 10;
             }
// перераховуємо max та min і зберігаємо відповідь:
        if( s > mx) { mx = s; x_max = X;}
        if( s < mi) { mi = s; x_min = X;}
    }
   cout <<  x_max << ' ' << x_min << '\n';
}
```

==========

Задача 58 (1095: Кількість Фібоначчі)

Серед даних чисел знайти кількість чисел Фібоначчі. Числами Фібоначчі називаються числа, перші два з яких дорівнюють одиниці, а кожне наступне рівне сумі двох попередніх. Наприклад: **1. 1, 2. 3, 5, 8, ...**

Формат вхідних даних.

У першому рядку задано число N $(1 \le N \le 10000)$. У наступному рядку ідуть самі цілі додатні числа не більші $2 \cdot 10^9$.

Формат вихідних даних:

У вихідний потік вивести кількість чисел Фібоначчі.

Приклад вхідних даних:

```
5
1 18 3 4 5
```

Приклад вихідних даних:

3

Програма мовою С++:

Ідея розв'язку: методом повного перебору знайдемо кількість чисел Фібоначчі у заданій послідовності.

```
#include <iostream>
using namespace std;
int main()
{
  int n, cnt = 0;
  cin >> n;
  while (  n-- )
    {
        long long x;
        cin  >> x;
// початкові заначення чисел Фібоначчі
        long long F1 = 1,F2 = 1,F = 1;
            while ( F < x )
            {
                F = F1 + F2;
                F1 = F2;
                F2 = F;
            }
// перевірка чи знайдене число є числом Фібоначчі
            if( F == x) cnt++;
    }
   cout <<  cnt << '\n';
}
```

Варіант 2. Оформимо фрагмент прорами перевірки чи належить число x послідовності Фібоначчі у вигляді функції:

```
#include <iostream>
using namespace std;
bool isfib(long long t)
{
    long long F1 = 1,F2 = 1,F = 1;
    while ( F < t )
      {
          F = F1 + F2;
          F1 = F2;
          F2 = F;
      }
    return (t == F);
}
int main()
```

```
{
  int n, cnt = 0;
  cin >> n;
  while (  n-- )
    {
        long long x;
        cin  >> x;
        if( isfib(x) ) cnt++;
    }
   cout <<  cnt << '\n';
}
```

==========

Задача 59 (1101: Максимальна сума дільників)

Дано число ***N***. Знайти число з проміжку від **1** до ***N*** з максимальною сумою дільників (включаючи не прості дільники, **1** і саме число). Якщо таких чисел декілька, то виведіть найменше з них.

Формат вхідних даних.

У стандартному потоці міститься ціле число N $(1 \le N \le 10^4)$

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

```
5
```

Приклад вихідних даних:

```
4
```

Програма мовою C++:

Ідея розв'язку: якщо число **i** є дільником числа **n**, то **n / i** – також дільник. Для чисел, що є повним квадратом, один і той же дільник **d = sqrt(n)** врахуємо двічі, а тому для таких чисел виконаємо окреме уточнення.

```
#include <iostream>
using namespace std;
int main()
{
  int n, x_max = 1, s_max = 1;
  cin >> n;
  for(int x = 2; x <= n; x++)
    {
        int s = x + 1, i = 2;
```

```
            for (  ; i * i < x; i++ )
                if( x % i == 0) s +=  i  +  ( x / i);

            if (i * i == x) s += i;
            if( s > s_max){ s_max = s; x_max = x;}
        }
      cout << x_max << '\n';
    }
```

==========

Задача 60 (1102: Скоротити дріб)

Скоротити дріб.

Формат вхідних даних.

В першому рядку вхідного потоку дано ***n*** - чисельник звичайного правильного дробу, а в наступному рядку ***m*** - його знаменник.

Формат вихідних даних:

У вихідний потік вивести скорочений дріб в такому ж форматі.

(1 ≤ *n*, *m* ≤ 2147483647)

Приклад вхідних даних:

```
5
10
```

Приклад вихідних даних:

```
1
2
```

Програма мовою C++:

Ідея розв'язку: якщо підключити математичну бібліотеку **<algorithm>**, то можна скористатися функцією **__gcd()**, яка реалізує алгоритм Евкліда знаходження найбільшого спільного дільника двох чисел.

```
#include <iostream>
#include <algorithm>
using namespace std;
int main()
{
    int a, b;
    cin >> a >> b;
    int d = __gcd(a,b);
    cout << a / d << '\n' << b / d << '\n';
}
```

У темі цикли, краще алгоритм Евкліда реалізувати, без використання вбудованої функції.

```
#include <iostream>
using namespace std;
int main()
{
    int a, b; cin >> a >> b;
    int n = a, m = b;
    while( m != 0)
    {
        n = n % m;
        swap(n,m);
    }
    cout << a / n << '\n' << b / n << '\n';
}
```

==========

Задача 61 (1103: Додавання дробів)

Виконайте додавання звичайних дробів. Результат виведіть у вигляді нескоротного дробу.

Формат вхідних даних.

В першому рядку вхідного потоку записано два натуральних числа n_1 і m_1, а у другому рядку n_2 і m_2. **(1 ≤ *n, m* ≤ 2147483647)**

Формат вихідних даних:

В один рядок вихідного потоку вивести чисельник і знаменник суми дробів через пропуск.

Приклад вхідних даних:

```
1 3
1 2
```

Приклад вихідних даних:

```
5 6
```

Програма мовою C++:

Ідея розв'язку: Сума двох дробів **a/b** та **c/d** є новий дріб чисельник якого рівний: **A = a * d + c * d**, а знаменник **B = b * d**. Знаменник та чисельник поділимо на найбільший спільний дільник **d**.

```
#include <iostream>
```

```
#include <algorithm>
using namespace std;
int main()
{
    long long  a, b, c, d;
    cin >> a >> b >> c >> d;
    long long A = a * d + c * b;
    long long B = b * d;
    long long D = __gcd(A,B);
    cout << A / D << ' ' << B / D << '\n';
}
```

==========

ТЕМА: заCHARованіSTRINGи

Задача 62 (1136: Кількість слів 2)

Дано рядок символів. Знайти кількість слів у даному рядку. Слова розділяються довільною кількістю пропусків.

Формат вихідних даних:

У стандартному потоці міститься рядок довжиною не більше **255** символів.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

I love you!

Приклад вихідних даних:

3

Програма мовою C++:

Ідея розвязку. Скористаємося тим фактом, що оператор зчитування **cin** повертає неявний параметр **true**, якщо зчитування з потоку пройшло успішно. Якщо при черговому звертанні до потоку, а він порожній, то неявний параметр приймає значення **false**. Наступний факт – оператор **cin** з потоку зчитує до першого пропуску.

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int k=0;
    string s;
    while ( cin >> s) k++;
    cout << k;
}
```

Примітка: на локальній машині цикл: **`while ( cin >> s)`** – це нескінчений цикл і перервати його можна комбінацією клавіш: **`Ctrl+Z`**.

==========

Задача 63 (1139: Паліндром)

Дано рядок символів. Перевірити чи є він паліндромом. Паліндромом називаються рядки, що однаково читаються зліва направо і справа наліво. Вивести «**`Yes`**» або «**`No`**».

Формат вихідних даних:

У стандартному потоці міститься рядок довжиною не більше **`255`** символів.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

`abba`

Приклад вихідних даних:

`Yes`

Програма мовою C++:

`Ідея розв'язку:` для рядка s скористаємося алгоритмом **`reverse(s.begin(),s.end())`**, який реалізований у бібліотеці **`<algorithm>`**

```
#include <algorithm>
using namespace std;
int main()
{
    string s;
    getline(cin,s);
    string t(s);
```

```
    reverse(s.begin(),s.end());
    if( t == s) cout << "Yes\n"; else cout << "No\n";
}
```

Варіант 2. Скористаємося методом двох вказівників. Встановимо перший вказівник на перший символ рядка, а другий вказівник – на крайній символ. Будемо рухати вказівники назустріч один одному, поки будуть рівними символи на які вони вказують.

```
#include <iostream>
using namespace std;
int main()
{
    string s;
    getline(cin,s);
    int n = s.size();
    int i = 0,j = n-1, flag = 1;
    while ( j - i > 0 )
    {
        if ( s[i] != s[j] ){ flag = 0; break;}
        i++;
        j--;
    }
   if( flag ) cout<<"Yes\n"; else cout << "No\n";
}
```

Примітка: рядки, які подає тестова система на перевірку, можуть містити пропуски, а тому слід скористатися зчитуванням рядка через `getline(cin,s)`.

==========

Задача 64 (11432: Пельмені для воїнів)

За допомогою пельменниці Дмитрик з мамою можуть зробити щонайбільше **X** пельменів за раз і на це треба **T** хвилин незалежно від кількості. Скільки часу займає виготовлення **N** пельменів?

Формат вхідних даних.

Вхідний потік містить цілі числа **N, X, T (1 < N, X,T < 1000)**

Формат вихідних даних:

У вихідний потік вивести шуканий час.

Приклад вхідних даних:

20 12 6

Приклад вихідних даних:

12

Приклад вхідних даних:

1000 1 1000

Приклад вихідних даних:

1000000

Програма мовою C++:

Ідея розв'язку: За **T** хв виготовляють **X** пельменів, тоді для виготовлення **N** пельменів необхідний час: $\left\lceil \frac{N}{X} \right\rceil \cdot T$. Округлення вверх : $\left\lceil \frac{N}{X} \right\rceil$ можна обчислити за формулою: **(N + X - 1) / X**. Цією формулою ми уникаємо команди розгалуження, бо обчислити загальний час можна і так:

```
k=N/X;
If (N % X != 0) k++;
```

```
#include <iostream>
using namespace std;
int main()
{
   int N,X,T;
   cin >> N >> X >> T;
   int k = (N+X-1)/X * T;
   cout << k;
}
```

==========

Задача 65 (11442: Максимальна відстань)

На площині є N точок, і-а з яких розташована на (x_i, y_i). Може бути кілька точок, які мають однакові координати. Яка максимальна манхетенська відстань між двома різними точками?

Манхетенська відстань між двома точками (x_i, y_i) і (x_j, y_j) визначається як $|x_i - x_j| + |y_i - y_j|$

Формат вхідних даних.

Перший рядок містить ціле число N $(2 <= N <= 2*10^5)$. Наступні **N** рядків містять цілі числа x_i, y_i $(1 \le x_i, y_i \le 10^9)$

Формат вихідних даних:

У вихідний потік виведіть шукану відстань.

Приклад вхідних даних:

```
3
1 1
2 4
3 2
```

Приклад вихідних даних:

```
4
```

Приклад вхідних даних:

```
2
1 1
1 1
```

Приклад вихідних даних:

```
0
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int M = 1000000005;
    int n; cin >> n;
    int mxS = -M, mx = - M;
    int miS = M, mi = M;

    for( int i = 0; i < n; i++)
    {
        int x, y; cin >> x >> y;
        miS = min(miS, x + y);
        mxS = max(mxS, x + y);
        mi = min(mi, x - y);
        mx = max(mx, x - y);
    }
    cout << max(mxS - miS, mx - mi) << endl;

}
```

==========

Задача 66 (11445: Кількість трійок)

Дано натуральне число N. Скільки трійок (A, B, C) натуральних чисел задовольняють $A \times B + C = N$?

Формат вхідних даних.

Вхідний потік містить ціле число N $(2 \leq N \leq 10^6)$

Формат вихідних даних:

У вихідний потік виведіть шукану кількість.

Приклад вхідних даних:

```
3
```

Приклад вихідних даних:

```
3
```

Приклад вхідних даних:

```
100
```

Приклад вихідних даних:

```
473
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
   int n; cin >> n;
   int k = 0;
   for(int i=1; i <= n-1; i++)
    k = k +(n - 1) / i;
   cout << k;
}
```

==========

Задача 67 (11451: Три точки на прямій)

Маємо N точок на координатній площині. i-а точка має координати (x_i, y_i).

Чи існує трійка різних точок, що лежать на одній прямій?

Формат вхідних даних.

Перший рядок містить ціле число N $(3 \le N \le 10^2)$. Наступні N рядків містять цілі числа x_i, y_i $(-10^3 \le x_i, y_i \le 10^3)$. Точки різні. Числа у рядках розділяються пропуском.

Формат вихідних даних:

У вихідний потік вивести **Yes** або **No** - відповідь на поставлене завдання

Приклад вхідних даних:

```
4
0 1
0 2
0 3
1 1
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

Ідея розв'язку: Для перевірки: чи лежать три точки, координати яких рівні **(x1, y1), (x2, y2), (x3, y3)**, на одній прямій то можна провести пряму через довільних дві точки, а координати третьої підставити у рівняння прямої: **(x3-x1)/(x2-x1)=(y3-y1)/(y2-y1)**. Це еквівалентно наступній рівності: **(x3-x1)·(y3-y1)=(x2-x1)·(y2-y1)**, яку легко закодити. У задачі здійснемо перевірку усіх трійок: якщо хоча б одна трійка точок належать деякій прямій, то відповідь ствердна. Дана задача є гарним прикладом показати використання користувацького типу: **struct** або використати пару: **pair<int, int>**.

```
#include <iostream>
using namespace std;
struct point
 {
     int x,y;
 };
 bool f ( point A, point B, point C)
 {
   return (B.x-A.x)*(C.y-A.y)-(C.x-A.x)*(B.y-A.y) == 0;
 }
int main()
{
    int n; cin >> n;
```

```
    point a[n];
    bool flag = false;
    for(int i = 0; i < n; i++) cin >> a[i].x >> a[i].y;
    for( int i = 0; i < n - 2; i++)
        for(int j = i + 1; j < n - 1; j++)
         for( int k = j + 1; k < n; k++)
           {
             if ( f(a[i],a[j],a[k]))
              {
                  i = k = j = n;
                  flag = true;
                  break;
              }
           }
     if(flag) cout << "Yes\n"; else cout << "No\n";
}
```

Варіант 2. Непоказний варіант без «зайвих типів» та функцій, але з двома масивами:

```
#include <iostream>
using namespace std;
int main()
{
 int n; cin >> n;
 int x[n],y[n];
 for (int i = 0; i < n; i++) cin >> x[i] >> y[i];
 for (int i = 0; i < n - 2; i++)
  for (int j = i + 1; j < n - 1; j++)
   for (int k = j + 1; k < n; k++)
    if ((x[j]-x[i])*(y[k]-y[j])==(x[k]-x[j])*(y[j]-y[i]))
       {
         cout << "Yes";
         return 0;
       }
  cout << "No";
}
```

==========

Задача 68 (1151: Розтин рядка)

Дано рядок символів довжиною не більше **255** символів, що містить один символ «-».

Формат вихідних даних:

У стандартному потоці міститься рядок довжиною не більше **255** символів.

Формат вихідних даних:

У першому рядку вихідного потоку вивести частину рядка до символу «**-**», у другому рядку - частину, що знаходиться після цього символу. Символ «**-**» не виводити в жодній частині.

Приклад вхідних даних:

Abba-Yes

Приклад вихідних даних:

Abba

Yes

Програма мовою C++:

Ідея розв'язку: знайдемо позицію входження символу **'-'** у рядок **s**, довжина якого **n**.

Довідка: функція **find('-')** – шукає позицію символу **'-'**, якщо такого символу немає, то дана функція повертає **-1**. Функція **s.substr(0,i)** – копіює підрядок, починаючи з першого символу і бере і символів. Якщо другого параметру немає, то копіюємо під рядок, починаємо з позиції **i+1** і до кінця рядка.

```
#include <iostream>
using namespace std;
int main(){
    string s;
    cin >> s;
    int i = s.find('-');
    cout << s.substr(0,i)<<'\n'<<s.substr(i+1)<<'\n';
}
```

Варіант 2: читаємо рядок і виводимо символи, поки не зустрінемо бар'єрний символ **'-'**, замість якого виводимо керуючий символ **'\n'**.

```
#include <iostream>
using namespace std;
int main(){
    string s;
    cin >> s;
    for ( char c: s)
        if( c == '-') cout << '\n';
          else cout <<c;
}
```

==========

Задача 69 (1152: Знайти суму)

Дано рядок виду: ***a#b=***, де ***a*** та ***b*** деякі цілі додатні числа не більші 10^4, а символ «#» - одна із операцій: «+», «-», «*». Знайти значення виразу ***s*** та у вихідний потік вивести рядок ***a#b=s***.

Формат вихідних даних:

У стандартному потоці міститься рядок довжиною не більше **255** символів.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

```
2+3=
```

Приклад вихідних даних:

```
2+3=5
```

Програма мовою C++:

Ідея розв'язку: у **C++** оператор зчитування **cin** наділений елементами штучного інтелекту, а це значить, що при зчитуванні він самостійно розділить входовий потік на типи, при умові, якщо відома структура входових даних.

```
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    char ch, op;
    cin >> a >> op >> b >> ch;
    int res = a + b;
    if(op == '-') res = a - b;
    if(op == '*') res = a * b;
    cout << a << op << b << ch << res << '\n';
}
```

==========

Задача 70 (1154: Сумуємо числа з рядка)

Дано рядок символів, що містить числа. У вихідний потік вивести суму цих чисел.

Формат вихідних даних:

У стандартному потоці міститься рядок довжиною не більше **255** символів. Числа та їх сума не перевищують $2 \cdot 10^9$.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

Abba1980 Yes5NO1990 0k2 5!

Приклад вихідних даних:

3982

Програма мовою C++:

Ідея розв'язку: Будемо переглядати символи рядка і вибирати усі підрядки, які містять тільки цифрові символи. Кожний такий підрядок переведемо у число і усі числа просумуємо. Оскільки рядок містить пропуски, то зчитуємо вхвдні дані командою: **getline(cin ,s);**

```
#include <iostream>
using namespace std;
int main()
{
   string s;
   getline( cin ,s);
   int n = s.size();
   int i = 0, sum = 0;
   while ( i < n)
   {
       if ( isdigit(s[i]))
        {
            string t ="";
            while ( i < n && isdigit(s[i]))
            {
                t = t + s[i];
                i++;
            }
            int x = stoi(t);
            sum += x;
        }
       else
            i++;
   }
   cout << sum;
}
```

==========

Рекурсія

Задача 71 (1165: n-й член послідовності)

Послідовність чисел задається таким рекурентним співвідношенням:

G(n)=2*G(n-2), G(0)=0, де G(1)=1. Для даного n знайти **G(n)**.

Формат вихідних даних:

У вхідному потоці дано ціле **n (0 < n < 100).**

Формат вихідних даних:

У вихідний потік вивести n-й член послідовності.

Приклад вхідних даних:

3

Приклад вихідних даних:

2

Програма мовою C++:

Ідея розв'язку: Рекурсивна функція **G(n)** задана рекурентно, а тому просто переносимо у код програми:

```
#include <iostream>
using namespace std;
long long G(int n)
     {
          if ( n == 0) return 0;
          if ( n == 1) return 1;
          return 2 * G(n - 2);
     }
int main()
{
  int a; cin >> a;
  cout << G(a);
}
```

==========

Задача 72 (1167: Дільники – Фібоначчі)

Знайти всі дільники числа ***N***, які є числами Фібоначчі.

Формат вихідних даних:

У вхідному потоці дано $N\ (1 < N < 10^6)$.

Формат вихідних даних:

У вихідний потік через пропуск вивести дільники у порядку зростання.

Приклад вхідних даних:

```
10
```

Приклад вихідних даних:

```
1 2 5
```

Програма мовою C++:

Ідея розв'язку: Усі дільники числа **n**, що є числами Фібоначчі, не перевищують половини числа, а тому перебираємо усі дільники числа до його половини.

```
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    int F1 = 1,F2 = 1,F = 1;//початкові значення чисел Фібоначчі
    while ( 2*F <= n)//поки дільник не перевищує  n/2
    {
        if ( n % F == 0) cout << F <<' ';
      // перехід до наступного числа Фібоначчі
        F = F1 + F2;
        F1 = F2;
        F2 = F;
    }
}
```

==========

Задача 73 (1170: Обернений порядок цифр)

Для заданого натурального числа N, вивести його цифри у зворотному порядку.

Формат вихідних даних:

У вхідному потоці задається натуральне N, що містить не більше **200** цифр.

Формат вихідних даних:

У вихідний потік вивести число з оберненим порядком цифр.

Приклад вхідних даних:

```
1230
```

Приклад вихідних даних:

0321

Програма мовою С++:

Ідея розв'язку: Рядок задамо у вигляді глобальної зміни **s**, для того щоб у рекурсивну функцію передавати лише довжину рядка. Рекурсивна функція зразу ж буде виводити символ рядка, а тому її тип буде **void** . Умова виходу із рекурсії – це нульове значення довжини рядка. Якщо довжина рядка не нуль, то виводимо поточний символ рядка і робимо рекурсивний виклик функції із значенням **n-1**.

```
#include <iostream>
using namespace std;
string s;
void G(int n)
{
    if ( n == 0) return;
    cout << s[n-1];
    G(n - 1);
}
int main()
{
  cin >> s;
  int a = (int)s.size();
  G(a);
}
```

==========

Задача 74 (1172: Біноміальні коефіцієнти)

Знайти біноміальний коефіцієнт для даних ***m, n*** (0 ≤ ***n*** ≤ ***m***), якщо

$$C(m,n) = \begin{cases} 1, & n=m,\ n=0,\ m<=1 \\ C(m-1,n-1) + C(m-1,n) \end{cases}$$

Формат вихідних даних:

У стандартному потоці міститься ***m, n*** (0 < ***m, n*** < 30).

Формат вихідних даних:

У стандартний потік вивести значення функції.

Приклад вхідних даних:

```
10 2
```

Приклад вихідних даних:

```
45
```

Програма мовою C++:

Ідея розв'язку: На основі трикутника Паскаля маємо формулу, яка дає можливість рекурентно задати обчислення біноміального коефіцієнта:

```
C(m, n) = C(m - 1, n - 1) + C(m - 1, n).
```

```
#include <iostream>
using namespace std;
int C(int m, int n)
{
    if ( n == 0 or n == m or m < 2 ) return 1;
    return C( m - 1, n - 1) + C(m - 1,n);
}
int main()
{
    int m,n; cin >> m >> n;
    cout << C(m,n);
}
```

Варіант 2: Для обмежень з умови задачі такий варіант пройде усі тести. Але він не ефективний. тому що, при занурені в рекурсію, виконуємо багато однакових обчислень. Щоб уникнути повторень, заведемо масив, у якому будемо зберігати значення біноміальних коефіцієнтів, які вже були обчислені раніше. Така рекурсія називається «лінивою» або рекурсія із запам'ятовуванням.

```
#include <iostream>
using namespace std;
long long dp[60][60] = {0};
long long C( int n, int k)
{
    if (k > n / 2) k = n - k; // правило симертрії
    if (k == 1)  return n;
    if (k == 0)  return 1;
    if (dp[n][k] == 0)
            dp[n][k] = C(n-1,k) +C (n - 1 ,k - 1);
    return dp[n][k];
}
int main()
{
    int m,n; cin >> m >> n;
```

```
        cout << C(m,n);
    }
```

==========

Задача 75 (1175: Чи є K?)

Дано ***N*** впорядкованих за зростанням натуральних чисел. Визначити чи є серед даних чисел число ***K***.

Формат вихідних даних:

У першому рядку вхідного потоку дано числа ***N, K*** **(0 < *N* < 10^6, 0 < *K* < 10^9).** Наступний рядок містить ***N*** чисел.

Формат вихідних даних:

У вихідний потік вивести "**Yes**" або "**No**" в залежності від наявності числа ***K***.

Приклад вхідних даних:

```
5 101
2 5 10 101 103
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

Ідея розв'язку: У рекурсивну функцію будемо передавати: **x** - число наявність якого в масиві, будемо визначати, **a[]** – цілочисельний масив, **n** – розмір масиву, **i** - поточний індекс елемента масиву. Наявність заданого числа будемо визначати по кількості його вжодження в масив.

```
#include <iostream>
using namespace std;
int Cnt(int x, int a[], int n, int i)
{
    if (i == n) return 0;//умова виходу з рекурсії
 // якщо поточний елемент рівний заданому, то до відповіді +1
    if (x == a[i]) return Cnt(x, a, n, i+1) + 1;
// в противному випадку переходимо до наступного елемента масива.
    return Cnt(x, a, n, i+1);
}

int main()
{
    int n,k; cin >> n >> k;
```

```
    int a[n];
    for(int i=0; i<n; i++) cin >> a[i];
    if( Cnt(k,a,n,0) != 0 ) cout << "Yes\n";
          else cout << "No\n";
}
```

==========

Задача 76 (1177: Старша цифра)

Напишіть програму, що виводить старшу цифру введеного натурального числа ***n***.

Формат вихідних даних:

У стандартному потоці міститься ***n*** $(0 < n < 10^9)$

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

2354

Приклад вихідних даних:

2

Програма мовою C++:

Ідея розв'язку: За умовою задачу вхідне число **x** матиме тип **int.** Умовою виходу із рекурсії є значення числа, яке має бути **x < 10** і рекурсивна функція його повертає. В противному випадку викликаємо рекурсію із числом, яке має значення **x/10**.

```
#include <iostream>
using namespace std;
int G(int x)
    {
        if ( x < 10 ) return x;
        return G( x / 10);
    }
int main()
{
    int a; cin >> a;
    cout << G(a);
}
```

==========

Множини

Задача 77 (1179: Сума перерізу множин)

Задано множини ***A*** та ***B***. Знайти суму тих елементів, що входять як у множину ***A***, так і в множину ***B***.

Формат вхідних даних.

У першому рядку знаходиться число ***N*** - кількість чисел множини ***A***. У наступному рядку міститься ***N*** чисел. Далі у новому рядку задається число ***M*** - кількість чисел множини ***B*** і у четвертому рядку містяться самі числа цієї множини. Всі числа додатні цілі та не перевищують **100**.

Формат вихідних даних:

У стандартний потік вивести суму чисел.

Приклад вхідних даних:

```
3
10 12 13
4
11 12 13 14
```

Приклад вихідних даних:

```
25
```

Програма мовою C++:

Ідея розв'язку: оскільки значення елементів множин не великі **(0 ≤ x < 101)**, то заведемо два масиви **a[101]** та **b[101]** типу **bool** в і-й комірці яких будемо зберігати true, якщо і належить множині. Наприклад, **a[54]=true** – це значить, що **54** є в множині **a**. Такий спосіб збереження даних називається сортування методом підрахунку.

```
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    bool a[101]{ false};// описуємо множину a
    // заповнюємо масив a
```

```
    for(int i=0; i<n; i++)
    {
        int x; cin >> x;
        a[x] = true;
    }
    int m; cin >> m;
    bool b[101]{false};// описуємо множину b
    // заповнюємо масив a
    for(int i=0; i < m; i++)
    {
        int x; cin >> x;
        b[x] = true;
    }
    int s = 0;
//якщо елемент i належить обом множинам, то його додаємо до суми s:
    for ( int i=0; i < 101; i++)
        if( a[i] &&  b[i]) s += i;
    cout << s << '\n';
}
```

==========

Задача 78 (1180: Сума об'єднання множин)

Задано множини ***A*** та ***B***. Знайти суму елементів, що входять в об'єднання цих множин.

Примітка. Об'єднання множин це елементи, що входять в множину A та в множину B і ніякі інші елементи.

Формат вихідних даних:

У першому рядку знаходиться число ***N*** - кількість чисел множини ***A***. У наступному рядку міститься ***N*** чисел. Далі у новому рядку задається число ***M*** - кількість чисел множини ***B*** і у четвертому рядку містяться самі числа цієї множини. Всі числа додатні цілі та не перевищуються **100**.

Формат вихідних даних:

У стандартний потік вивести суму чисел.

Приклад вхідних даних:

```
3
10 12 13
4
```

11 12 13 14

Приклад вихідних даних:

60

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n, s = 0; cin >> n;
    int a[101]{0};
    for( int i = 0; i < n; i++)
    {
        int x; cin >> x;
        if (a[x] == 0)
         {
            s = s + x;
            a[x] = 1;
         }
    }
    int m; cin >> m;
    for( int i = 0; i < m; i++)
    {
        int x; cin >> x;
        if (a[x] == 0)
         {
            s = s + x;
            a[x] = 1;
         }
    }
    cout << s << endl;
}
```

==========

Задача 79 (1183: Відсутні літери)

Дано два слова, що складаються з малих латинських літер. Вивести ті літери, що не входять в жодне із слів.

Формат вихідних даних:

В першому рядку знаходиться перше слово, у другому - друге.

Формат вихідних даних:

У вихідний потік вивести відсутні малі літери латинського алфавіту.

Приклад вхідних даних:

qwertyuiop

asdfghjkl

Приклад вихідних даних:

bcmnvxz

Програма мовою C++:

Ідея розв'язку: ідею запозичимо із задачі(1179: Сума перерізу множин):

```
#include <iostream>
using namespace std;
int main()
{
    string sa,sb;
    cin >> sa >> sb;
    string t = ”“;
    bool a[27]{false};
    for( char c : sa)
    {
        int x = c - 96;
        a[x] = true;
    }
    bool b[27]{false};
    for( char c: sb)
    {
        int x = c - 96;
        b[x] = true;
    }
    string s = "";
    for ( int i = 1; i < 27; i++)
        if( !a[i] &&  !b[i]) s += (char)(i + 96);
    cout << s << '\n';
}
```

Варіант 2: скористаємося функцією пошуку **find()**, яка повертає індекс першого входження символу в рядок, або повертає **-1**, якщо символ в рядку відсутній. Цикл організуємо по латинській абетці і для кожної літери викликаємо **find()** для рядків **sa** та **sb**.

```
#include <iostream>
using namespace std;
int main()
{
    string sa,sb,s = "";
    cin >> sa >> sb;
    for( char c ='a'; c <= 'z'; c++)
    {
```

```
            int i = sa.find(c);
            int j = sb.find(c);
            if( i == -1 && j == -1) cout << c;
        }
}
```

==========

Задача 80 (1184: Більше двох разів)

Заданий текст, що складається з малих латинських літер. Вивести ті літери, що зустрічаються в тексті не менше двох разів.

Формат вихідних даних:

У вхідному потоці знаходиться текст довжиною не більше **255** символів.

Формат вихідних даних:

Вивести в алфавітному порядку через пропуск літери, що зустрічаються не менше двох разів.

Приклад вхідних даних:

```
the essence of
```

Приклад вихідних даних:

```
e s
```

Програма мовою C++:

Ідея розв'язку: скористаємося методом підрахунку (див. задачу 1179)

```
#include <iostream>
using namespace std;
int main()
{
    string sa;
    getline(cin, sa);
    int a[27]{0}; // для кожної літери заведемо лічильник
    for( char c: sa)
    {
      if(isalpha(c)) // якщо символ с – літера то
      {
      int i = c - 96; // по коду отримуємо індекс елемента масиву
      a[i]++; // відповідний елемент масиву збільшуємо на 1
      }
    }
    for ( int i = 1; i < 27;i++) // перебираємо елементи масиву
        if ( a[i] > 1) // якщо
         {
```

```
            char c = (char)(i + 96); //отримуємо код літери
            cout << c << ' '; // вивід літери
        }
}
```

==========

Задача 81 (1188: Відсутні цифри)

Задані ***N*** натуральних чисел. Для кожного введеного числа надрукувати в порядку зростання усі цифри, що не входять в десятковий запис цього числа.

Формат вихідних даних:

У першому рядку міститься число ***N*** **(1 < *N* < 1000).**

У наступних ***N*** рядках знаходяться десяткові записи цілих чисел. Кількість цифр чисел не перевищує **100**.

Формат вихідних даних:

Вивести у ***N*** рядках цифри, що відповідають умові задачі. Якщо таких цифр немає, то виводити порожній рядок.

Приклад вхідних даних:

```
1
16
```

Приклад вихідних даних:

```
02345789
```

Програма мовою C++:

Ідея розв'язку:

```
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    string sa;
    while ( n-- )
    {
        cin >> sa;
        for ( char c = '0'; c <= '9'; c++)
            if((int)sa.find(c) == -1 ) cout << c;
        cout << '\n';
    }
```

```
}
```

==========

Задача 82 (1192: Різниця: max-min)

Знайти різницю між найбільшим та найменшим числом.

Формат вихідних даних:

У стандартному потоці містяться числа по одному у рядку. Кількість чисел не більше 10^5. числа не перевищують по модулю 10^4.

Формат вихідних даних:

У стандартний потік вивести результат.

Приклад вхідних даних:

```
3
6
1
11
```

Приклад вихідних даних:

```
10
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n; cin >> n;
    int mx = -20000, mi = 20000,x;
    while (cin >> x)
    {
        mx = max(x,mx);
        mi = min(x,mi);
    }
    cout << mx - mi << '\n';
}
```

==========

Задача 83 (1193: Найдовше слово)

Знайти найдовше слово.

Формат вихідних даних:

У стандартному потоці задані слова по одному у кожному рядку. Довжина слів не перевищує **255** символів, кількість слів не більше 10^4.

Формат вихідних даних:

У стандартний потік вивести найдовше слово, що першим зустрілося у переліку.

Приклад вхідних даних:

```
abba
words
files
```

Приклад вихідних даних:

```
words
```

Програма мовою C++:

Ідея розв'язку:

Скористаємося тим, що функція **cin** з потоку бере рядок до першого пропуску. Отже поки потік не порожній будемо брати слово і перераховувати поточний максимум, який відповідає довжині слово.

```
#include <iostream>
using namespace std;
int main()
{
    int len = 0;
    string t, smax = "";
    while (cin >> t)
    {
        int n = t.size();
        if( n > len)
        {
            len = n;
            smax = t;
        }
    }
    cout << smax << '\n';
}
```

==========

Геометрія

Задача 84 (1196: Вершини трикутника?)

Задані координати трьох точок на площині. Визначити чи можуть вони бути вершинами трикутника.

Формат вихідних даних:

У стандартному потоці задані через пропуск координати трьох точок **x_1, y_1, x_2, y_2, x_3 ,y_3**

Формат вихідних даних:

У стандартний потік вивести **Yes** коли такий трикутник існує і **No** в іншому випадку.

Приклад вхідних даних:

```
0 0 1.5 0 1 1
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

Ідея розв'язку:

```
#include <iostream>
using namespace std;
int main()
{
   double x1,y1,x2,y2,x3,y3;
   cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
   double d = (x2 - x1)*(y3 - y1)-(x3 - x1)*(y2 - y1);
   if(d != 0) cout <<"Yes\n"; else cout <<"No\n";
}
```

Варіант 2:

Скористаємося умовою трикутника, який заданий своїми сторонами: якщо сума двох менших сторін більша за найбільшу сторону, то такий трикутник існує. Сторона трикутника це довжина відрізка, який заданий своїми координатами і обчислюється за формулою: **$d = sqrt((x1-x2)^2-(y1-y2)^2)$.**

```
#include <iostream>
#include <cmath>
```

```
using namespace std;
double d ( double x1, double y1, double x2, double y2)
{
 return sqrt((x1 - x2) * (x1 - x2) + (y1- y2) *(y1-y2));
}
int main()
{
   double x1,y1,x2,y2,x3,y3;
   cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
   double a =d(x1, y1, x2, y2);
   double b =d(x2, y2, x3, y3);
   double c =d(x3, y3, x1, y1);
   if( a > b) swap(a, b);
   if( b > c) swap(b, c);
   if(a + b > c) cout <<"Yes\n"; else cout <<"No\n";
}
```

==========

Задача 85 (1197: Найбільша сума цифр)

Задано ***N*** натуральних чисел. Знайти число з найбільшою сумою цифр.

Формат вихідних даних:

У стандартному потоці дано N $(1 < N < 10^5)$.

У наступному рядку через пропуск задаються самі додатні числа менші **32000**.

Формат вихідних даних:

У стандартний потік вивести перше з чисел з найбільшою сумою цифр.

Приклад вхідних даних:

```
4
31 5 11 25
```

Приклад вихідних даних:

```
25
```

Програма мовою C++:

Ідея розв'язку: для спрощення коду напишемо функцію, яка визначає суму цифр числа, а далі перебираємо усі числа з потоку і якщо його сума більша за поточну, то переписуємо відповідь.

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

```
int sumd ( int x)
{
    int s = 0;
    while ( x != 0)
    {
        s = s + ( x % 10);
        x = x / 10;
    }
    return s;
}
int main()
{
   int mx = 0, smx = 0,n;
   cin >> n;
   while ( n--)
   {
       int a; cin >> a;
       int t = sumd(a);
       if ( smx < t)
       {
           smx = t;
           mx = a;
       }
   }
  cout << mx << '\n';
}
```

==========

Задача 86. (1198: Найбільша сума дільників)

Задано ***N*** натуральних чисел. Знайти число з найбільшою сумою дільників.

Формат вихідних даних:

У стандартному потоці дано ***N*** **(1 ≤ *N* ≤ 1000).** У наступному рядку через пропуск задаються самі числа.

Формат вихідних даних:

У стандартний потік вивести перше з чисел з найбільшою сумою дільників.

Приклад вхідних даних:

```
4
31 5 11 25
```

Приклад вихідних даних:

```
31
```

Програма мовою C++:

Ідея розв'язку: основна частина програми така ж, як у попередній задачі, інша функція, яка шукає суму дільників. На початку сума рівна сумі самого числа плюс одиниця – як дільники будь якого числа. Починаємо з 2 і перебір дільників здійснюємо до кореня квадратного із числа. Якщо і – дільник числа х, то х/і також дільник і на ці два числа збільшується сума. Окремого уточнення потребує число, що є повним квадратом.

```
#include <iostream>
#include <cmath>
using namespace std;
long long sumd ( long long x)
{
    long long s = x + 1,i = 2;
    for ( ; i*i <= x; i++)
      if ( x % i == 0) s = s + i + x / i;
    if ( i*i == x) s += i;
    return s;
}
int main()
{
   int mx = 0, smx = 0,n;
   cin >> n;
   while ( n--)
   {
       long long a; cin >> a;
       int t = sumd(a);
       if ( smx < t)
       {
           smx = t;
           mx = a;
       }
   }
  cout << mx << '\n';
}
```

==========

Задача 87. (1199: Точки перетину кіл)

Дано два різні кола. Визначити кількість точок їх перетину.

Формат вихідних даних:

У стандартному потоці задані цілі числа x_1, y_1, r_1, x_2, y_2, r_2.

Кожне із чисел не перевищують по модулю **30000**, радіуси кіл – додатні.

Формат вихідних даних:

У стандартний потік вивести кількість точок перетину кіл або **0** у випадку коли кола не перетинаються.

Приклад вхідних даних:

```
0 0 5 10 0 5
```

Приклад вихідних даних:

```
1
```

Програма мовою C++:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    double x1,y1,r1,x2,y2,r2;
    cin >> x1 >> y1 >> r1 >> x2 >> y2 >> r2;
    double d = sqrt(pow(x1 - x2, 2)+ pow(y1 - y2, 2));
    int ans = 2;
    if (d > r1 + r2 || d < fabs(r1 - r2)) ans = 0;
      else
        if(d == r1 + r2 || d == fabs(r1 - r2)) ans = 1;
    cout << ans << endl;
}
```

==========

Задача 88. (1201: Найбільш віддалені точки)

На площині задано **N** точок з цілими координатами. Знайти дві найбільш віддалені точки. Якщо таких пар є декілька, то вивести ті, що мають менші порядкові номери. Перша виведена точка повинна мати завжди менший номер, ніж друга.

Формат вихідних даних:

У стандартному потоці перший рядок містить ціле N $(0 < N < 1000)$.

У наступних N рядках міститься по два числа, які розділяються пропуском – координати точок X_i, Y_i $(0 \le X_i, Y_i \le 10^5)$.

Формат вихідних даних:

У стандартний потік вивести в одному рядку два числа – порядкові номери точок, відстань між якими є найбільшою.

Приклад вхідних даних:

```
3
0 0
10 0
-10 -10
```

Приклад вихідних даних:

```
2 3
```

Програма мовою C++:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n; cin >> n;
    pair <int, int> a[n];
    for(int i = 0; i < n; i++)
       cin >> a[i].first >> a[i].second;
    double mx = 0;
    int id1 = 0, id2 = 0;
    for( int i = 0; i < n-1; i++ )
        for ( int j = i + 1; j < n; j++)
        {
            long long X = (a[i].first - a[j].first);
            long long Y = (a[i].second - a[j].second);
            double dx = sqrt(1.0 * X * X + 1.0 * Y * Y);
            if ( dx > mx)
            {
                mx = dx;
                id1 = i;
                id2 = j;
            }
        }
     cout << id1 + 1 << ' ' << id2 + 1 << endl;
}
```

==========

Проста математика

Задача 89. (1203: Найближча за площею)

Дано назви геометричних фігур та їх площу. Знайти фігуру із площею, що є найближчою до площі першої у списку фігури. Якщо таких є декілька, то вивести ту, що йде у переліку раніше.

Формат вихідних даних:

У першому рядку вхідного потоку міститься ціле число N $(1 \le N \le 1000)$ – кількість рядків даних.

Далі у наступних $2 \cdot N$ рядках містяться в окремих рядках назва фігури та її площа. Назва фігури не перевищує 30 символів і складається з малих латинських літер, а площа є дійсним числом.

Формат вихідних даних:

У вихідний потік вивести в одному рядку назву фігури і через пропуск її площу.

Приклад вхідних даних:

```
3
kolo
2.52
figura
10.12
kolo
3.25
```

Приклад вихідних даних:

```
kolo 3.25
```

Програма мовою C++:

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n; cin >> n;
    string s, st, t;
    double a, d, x;
```

```
    cin >> s >> d;
    cin >> st >> x;
    double len = fabs( d - x);
    for(int i = 2; i < n; i++)
        {
            cin >> t >> a;
            if ( fabs( d - a) < len)
            {
                len = fabs(d - a);
                st = t;
                x = a;
            }
        }

    cout << st  << ' ' << x << "\n";
}
```

==========

Задача 90. (1207: Середні бали)

Дано список прізвищ ***N*** учнів з їх оцінками по ***K*** предметах. Знайти середні бали для кожного учня.

Формат вихідних даних:

У першому рядку вхідного потоку дано цілі числа ***N, K*** **(1 ≤ *N* ≤ 1000, 1 ≤ *K* ≤ 10).** Далі у ***2 N*** рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться ***K*** оцінок. Прізвища довжиною не більше **30** символів.

Формат вихідних даних:

У вихідний потік вивести в окремих рядках прізвища учнів і їх середні бали з одним знаком після коми.

Приклад вхідних даних:

```
3 4
Tarasenko
4 4 5 5
Sydorenko
2 3 3 2
Pochatenko
```

```
5 5 5 5
```

Приклад вихідних даних:

```
Tarasenko 4.5
Sydorenko 2.5
Pochatenko 5.0
```

Програма мовою С++:

Ідея розв'язку задачі: задача на моделювання ситуації, яка описана в умові задачі.

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

int main()
{
  int n, k;
  cin >> n >> k;
  while ( n--)// поки є дані в потоці
  {
       string s;
       cin >> s; // читаємо прізвище
       int sum =0; // початкова сума балів
       for(int i = 0; i < k; i++)
       {
            int x; cin >> x; // читаємо з потоку поточний бал
            sum += x; // накопичуємо суму
       }
       double averge = 1.0 * sum / k; // обчислення середнього
балу
    // вивід результату з одним знаком після коми
     cout<<s<<' '<<fixed <<setprecision(1)<<averge<<'\n';

  }
}
```

==========

Задача 91. (1208: Максимальний середній бал)

Дано список прізвищ ***N*** учнів з їх оцінками по ***K*** предметах. Вивести список учнів, що мають максимальний середній бал.

Формат вихідних даних:

У першому рядку вхідного потоку дано цілі числа ***N, K*** **(1 ≤ *N* ≤ 1000, 1 ≤ *K* ≤ 10).**

Далі у ***2 N*** рядках дані розміщені таким чином: спочатку в окремому рядку іде прізвище, а у наступному через пропуск знаходяться ***K*** оцінок. Прізвища довжиною не більше 30 символів.

Формат вихідних даних:

У вихідний потік вивести в окремих рядках прізвища учнів, які мають максимальний середній бал.

Приклад вхідних даних:

```
3 4
Tarasenko
4 4 5 5
Petrenko
2 3 3 2
Sydorenko
5 5 5 5
```

Приклад вихідних даних:

```
Sydorenko
```

Програма мовою C++:

Ідея розв'язку задачі: програма подібна до попередньої (1207) тільки додатково будемо підтримувати максимальне значення, на якому отримуємо максимальний середній бал. Слід зауважити, що для визначення максимального середнього слід підтримувати максимальну суму.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
  int n,k, mx = 0;
  string s,t;
  cin >> n >> k;
  while ( n--)
  {
```

```
        cin >> s;
        int sum =0;
        for(int i = 0; i < k; i++)
        {
            int x; cin >> x;
            sum += x;
        }
        if( sum > mx)
           {
            t = s;
            mx = sum;
           }

   }
    double avrg = 1.0 * mx / k;
    cout << t <<' '<<fixed << setprecision(1)<<avrg<<'\n';

}
```

==========

Задача 92. (1377: Порівняти числа)

Васильку на гуртку з програмування дали завдання написати програму для порівняння чисел a^n та b^n. Програму він написав швидко, але не всі тести у нього проходять успішно. Допоможіть йому вірно написати таку програму.

Формат вхідних даних.

Перший рядок вхідного потоку містить T – кількість тестів. Наступні T рядків містять по три цілі числа: a, b, n.

Обмеження: $1 \le T \le 1000$; $|a|, |b| \le 10^9$; $1 \le n \le 10^9$.

Формат вихідних даних:

Для кожного тесту в окремому рядку вивести:

1, якщо $a^n > b^n$;

2, якщо $a^n < b^n$;

0, якщо $a^n = b^n$.

Приклад вхідних даних:

```
2
3 4 5
-3 2 4
```

Приклад вихідних даних:

```
2
1
```

Програма мовою C++:

```
#include <iostream>
using namespace std;
int main()
{
  int test;
  cin >> test;
  while ( test--)
  {
    int a, b, n, res = 1;
    cin >> a >> b >> n;
    if( n % 2 == 0)
        {
          if( abs(b) > abs(a) ) res = 2;
        }
    else
        {
            if ( b > a) res = 2;
        }
    cout << res << '\n';
  }
}
```

==========

Задача 93. (1218: Відсоток)

Васильку, як юному математику, вчителька доручила визначити процент учнів, що мають бал вищий за середній у класі. Спробуйте йому допомогти в цьому; складіть відповідну програму.

Формат вихідних даних:

У першому рядку стандартного вхідного потоку міститься кількість учнів в класі ***N (N ≤ 100).*** Наступний рядок містить **N** цілих чисел не більших **100** – оцінки учнів по **100** - бальній шкалі.

Формат вихідних даних:

У стандартний вихідний потік вивести шуканий процент з трьома знаками після коми.

Приклад вхідних даних:

```
9
100 99 98 97 96 95 94 93 91
```

Приклад вихідних даних:

```
55.556%
```

Програма мовою C++:

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    int n; cin >> n;
    int a[n], sum = 0;
    for(int i = 0; i < n; i++)
        {
            cin >> a[i];
            sum += a[i];
        }
    int k = 0;
    for( int i = 0; i < n; i++)
        if ( n * a[i] > sum) k++;
    double p = 100.0 * k / n;
    cout << fixed << setprecision(3) << p << "%\n";
}
```

==========

Задача 94.

(1256: Кількість дільників числа у канонічному розкладі)

Потрібно визначити, скільки дільників має натуральне число, подане в канонічному розкладі?

Формат вхідних даних.

У першому рядку дано натуральне число ***n***, що показує кількість простих множників у канонічному розкладі **(1 ≤ *n* ≤ 20).**

У наступних ***n*** рядках дано по два натуральних числа, що не перевищують **100**: простий дільник та його кількість входження у канонічний розклад даного числа.

Формат вихідних даних:

Виведіть одне натуральне число - кількість дільників цього числа (відповідь гарантовано не перевищує 10^{18}).

Приклад вхідних даних:

```
2
2 2
3 1
```

Приклад вихідних даних:

```
6
```

Програма мовою C++:

Ідея розв'язку: основна теорема арифметики гласить: кожне натуральне число n єдиним чином подається у вигляді добутку простих чисел:

$$n = p_1^{d1} * p_2^{d2} * \ldots * p_n^{dn}$$

Кількість дільників буде визначатися за формулою:

$$d = (1 + d_1) * (1 + d_2) * \ldots * (1 + d_n).$$

На основі цієї формули і будуємо наш алгоритм:

```
#include <iostream>
using namespace std;
int main()
{
  int n; cin >> n;
  long long cnt = 1;
  while ( n--)
  {
      int p, d;
      cin >> p >> d;
      cnt = cnt *(1 + d);
  }
  cout << cnt <<'\n';
```

```
}
```

==========

Задача 95. (1261: Холодильник)

Розміри холодильника ***A × B × C***. Чи можливо перенести його через дверний проріз розміром ***X*** × ***Y***?

Вважатимемо, що холодильник можна нести тільки під такими кутами, щоб деякі дві сторони холодильника були паралельні сторонам прорізу.

Формат вхідних даних.

У стандартному потоці міститься п'ять чисел ***A, B, C, X, Y*** **(1 ≤ *A, B, C, X, Y* ≤ 100).**

Формат вихідних даних:

У стандартний потік вивести слово **YES**, якщо холодильник можна перенести через дверний проріз, і **NO** – якщо ні.

Приклад вхідних даних:

```
4 5 6 10 20
```

Приклад вихідних даних:

```
YES
```

Програма мовою C++:

Ідея розв'язку: для зменшення кількості умов перевірок відсортуємо розміри холодильника у порядку не спадання (у реальних умовах це відповідає вибору найменших сторін), а також перезначимо назви сторін дверей так щоб **x** – найменша сторона. Після цих перетворень умова проходження буде така: **a <= x && b <= y.** Чому нестрога нерівність – не знаю (авторська ізюминка).

```
#include <iostream>
using namespace std;
int main()
{
  int a,b,c,x,y;
  cin >> a >> b >> c >> x >> y;
  if( a > b) swap(a, b);
```

```
  if( b > c) swap(c, b);
  if( a > b) swap(a, b);
  if( x > y) swap(x, y);
  if( a <= x && b <= y) cout <<"YES\n"; else cout <<"NO\n";
}
```

==========

Задача 96. (1265: Прямокутник і точка)

На площині координатами своїх діагональних кутів задано прямокутник, сторони якого паралельні осям координат. Крім того, на тій самій площині задано точку. Потрібно визначити, чи попадає задана точка в прямокутник.

Формат вхідних даних.

Перший рядок вхідного потоку містить 4 цілих числа - координати діагональних кутів прямокутника. Другий рядок містить ще два цілих числа – координати точки. Кожне з чисел не перевищує по абсолютній величині 10000.

Формат вихідних даних:

Єдиний рядок вихідного потоку повинен містити слово «**Yes**», якщо точка лежить всередині прямокутника, та «**No**» у протилежному випадку. Якщо точка лежить на стороні прямокутника, вважається, що вона лежить всередині прямокутника.

Приклад вхідних даних:

```
0 100 100 0 50 50
```

Приклад вихідних даних:

```
Yes
```

Програма мовою C++:

Ідея розв'язку:

Якщо точка належить прямокутнику, то її координати мають задовольняти такі умови: **min(x1,x2)<= x && x <= max(x1,x2) та min(y1,y2)<= y && y <= max(y1,y2).**

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int x1, y1, x2,y2,x,y;
    cin >> x1 >> y1 >> x2 >> y2 >> x >> y;
    int A = min(x1,x2)<= x && x <= max(x1,x2);
    int B = min(y1,y2)<= y && y <= max(y1,y2);
    if( A && B) cout <<"Yes\n"; else cout <<"No\n";
}
```

Варіант 2:

Вершини прямокутника і координати точки будемо зберігати у `pair< int, int >`. Пара сортується по першому параметрі, а при умові їх рівності – по другому.

```
#include <iostream>
using namespace std;
int main().
{
    int x, y; cin >> x >> y;
    pair< int,int > p1{x,y};
    cin >> x >> y;
    pair< int,int > p2{x,y};
    cin >> x >> y;
    pair< int,int > p{x,y};
    if (p1 > p2) swap(p1,p2);
    if(p1.second > p2. second) swap(p1.second,p2.second);
    if( p >= p1 && p <= p2) cout <<"Yes\n";
      else cout <<"No\n";
}
```

==========

Задача 97. (1285: Розподіл яблук порівну)

N школярів розподіляють ***K*** яблук «порівну», тобто таким чином, щоб кількість яблук, що отримали два будь-яких школярі відрізнялася не більше ніж на **1**. Напишіть програму, яка визначає кількість школярів, яким дісталося яблук менше, ніж деяким їхнім товаришам.

Формат вхідних даних.

З вхідного потоку вводиться два цілих числа N $(1 \le N \le 100)$ та K $(1 \le K \le 10^5)$.

Формат вихідних даних:

У вихідний потік необхідно вивести єдине ціле число – кількість школярів, яким дісталося яблук менше, ніж деяким їхнім товаришам.

Приклад вхідних даних:

```
7 30
```

Приклад вихідних даних:

```
5
```

Програма мовою C++:

Ідея розв'язку: У випадку коли n є дільником **k**, то відповідь **0**, у противному випадку відповідь буде рівна n – **(k % n)**.

```
#include <iostream>
using namespace std;
int main()
{
   int n, k;
   cin >> n >> k;
   int b = k % n;
   if ( b == 0) b = n;
   cout << n - b;
}
```

==========

Задача 98. (1288: Різниця часу)

Один годинник показує H_1 годин M_1 хвилин S_1 секунд, а інший – H_2 годин M_2 хвилин S_2 секунд. Напишіть програму, що визначає на скільки годин, хвилин та секунд відрізняються показання годинників.

Формат вхідних даних.

З вхідного потоку спочатку вводиться три цілих числа H_1 $(0 \le H_1 \le 23)$, M_1, S_1 $(0 \le M_1, S_1 < 60)$. А потім ще три цілих числа H_2 $(0 \le H_2 \le 23)$, M_2, S_2 $(0 \le M_2, S_2 < 60)$.

Формат вихідних даних:

У вихідний потік необхідно вивести три натуральних числа, що розділені пробілом різницю часу у годинах, хвилинах та секундах. Перше число повинно

бути в діапазоні від **0** до **23** включно, друге (хвилини) та третє (секунди) числа повинні бути з діапазону від **0** до **59** включно.

Приклад вхідних даних:

```
3 15 50
3 16 0
```

Приклад вихідних даних:

```
0 0 10
```

Програма мовою C++:

Ідея розв'язку: у подібних задачах слід дотримуватися такого правила: якщо вхідні дані подані у різних одиницях виміру, то слід ці дані звести до однієї системи виміру.

```
#include <iostream>
using namespace std;
int main()
{
    int h,m,s;
    cin >> h >> m >> s;// читаємо покази першого годинника
    int x = h * 3600 + m * 60 + s; // загальні секунди
    cin >> h >> m >> s; ;// читаємо покази другого годинника
    int y = h * 3600 + m * 60 + s; // загальні секунди
    x = abs(y - x);// різниця показів
// переводимо секунду у години, хвилини та секунди
    h = x / 3600;
    x = x % 3600;
    m = x / 60;
    s = x % 60;
    cout << h << ' ' << m << ' ' << s;
}
```

Задача 99 (1387: Підпослідовність з одиниць)

У нас є рядок **S**. Кожен символ цього рядка дорівнює **0** або **1**. Перевірте чи всі одиниці в цьому рядку утворюють єдиний непорожній відрізок (підпослідовність) послідовності.

Формат вхідних даних:

Перший рядок вхідного потоку містить ціле число **T (1 ≤ T ≤ 10)** - кількість тестів. Наступні **T** рядків містять тести - єдиний рядок **S (1 ≤ |S| ≤ 10^5)**

Формат вихідних даних:

Для кожного тесту виведіть **YES** або **NO** - відповідь на поставлену задачу.

Приклад вхідних даних:

```
6
001111110
00110011
000
1111
101010101
101111111111
```

Приклад вихідних даних:

```
YES
NO
NO
YES
NO
NO
```

Ідея розв'язку: знайдемо позицію першої одиниці. Якщо в результаті перевірки не знайдемо жодної, то прапорець перекинимо в **0** – це значить, що відповідь **NO**. Далі знайдемо позицію першої одиниці рухаючись з кінця. Якщо між цими позиціями знайдеться хоча б один нуль, то прапорець перекидаємо в **0**. В залежності від прапорця виводимо відповідь:

```
#include <iostream>
using namespace std;
int main()
{
  int test;
  cin >> test;
  cin.ignore();
  while ( test--)
  {
     string s; cin >> s;
     int n = s.size(), flag = 1;;
     int i = 0;
     while (s[i] == '0' && i < n) i++;
     if(i == n) flag = 0;
```

```
        else
            {
                int j = n-1;
                while( s[j] == '0') j-- ;
                for ( int k = i; k <= j; k++)
                    if( s[k] == '0') flag = 0;
            }
        if ( flag) cout << "YES\n"; else cout << "NO\n";
    }
}
```

Зміст

Пам'ятка для вчителя.

Даний посібник є додатком до електронного ресурсу «Школа олімпійського резерву» [доступ: https://sbs.hoippo.km.ua/]. Вчителі можуть реєструвати школи, самі реєструватися, пропонувати реєструватися учням. У посібнику розглянуті лише 99 задач, а на даному ресурсі таких задач близько 900. Автоматизація перевірки розв'язків дає можливість якісно вивчати сучасні мови програмування.