Управління освіти і науки
Чернігівської обласної державної адміністрації

Чернігівський обласний інститут післядипломної
педагогічної освіти імені К.Д. Ушинського

ЗБІРНИК ЗАДАЧ ТА РОЗВ'ЯЗКІВ ІІІ ЕТАПУ ВСЕУКРАЇНСЬКОЇ УЧНІВСЬКОЇ ОЛІМПІАДИ З ІНФОРМАТИКИ

2013-2014 НАВЧАЛЬНОГО РОКУ

Чернігів – 2015

Збірник задач та розв'язків ІІІ етапу Всеукраїнських учнівських олімпіад з інформатики 2013-2014 навчального року /укл. О.Є. Баранова, Ю.М. Літош, В.В. Зуб, С.М. Бондаренко, О.М. Смірнова. – Чернігів: ЧОІППО імені К.Д. Ушинського, 2015. – 60 с.

Укладачі:

Баранова О.Є., методист відділу інформатики, інформаційних технологій та дистанційного навчання Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського

Літош Ю.М., завідувач відділу інформатики, інформаційних технологій та дистанційного навчання Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського

Зуб В.В., директор Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учитель-методист

Бондаренко С.М., учитель математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учитель-методист

Смірнова О.М., методист відділу інформатики, інформаційних технологій та дистанційного навчання Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського

Рецензенти:

Горошко Ю.В., завідувач кафедри інформатики та обчислювальної техніки Чернігівського національного педагогічного університету імені Т.Г. Шевченка, доктор педагогічних наук

Покришень Д.А., завідувач кафедри інформатики та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського, кандидат педагогічних наук, доцент

Рекомендовано до друку
вченою радою Чернігівського обласного інституту післядипломної педагогічної освіти імені К.Д. Ушинського (протокол №6 від 17.09.2015р.)

ЗМІСТ

8-11 клас

I тур

A - Кондиціонер Степана

Input file name:	cond.in
Output file name:	cond.out
Time limit:	100 ms
Memory limit:	256 M

В офісі, де Степан працює програмістом, установили кондиціонер нового типу. Цей кондиціонер відрізняється особливою простотою в управлінні. У кондиціонера є всього лише два керованих параметри: бажана температура і режим роботи.

Кондиціонер може працювати в наступних чотирьох режимах:

- «freeze» - охолодження. У цьому режимі кондиціонер може тільки зменшувати температуру. Якщо температура в кімнаті і так не більше бажаної, то він вимикається.
- «heat» - нагрів. У цьому режимі кондиціонер може тільки збільшувати температуру. Якщо температура в кімнаті і так не менше бажаної, то він вимикається.
- «auto» - автоматичний режим. У цьому режимі кондиціонер може як збільшувати, так і зменшувати температуру в кімнаті до бажаної.
- «fan» - вентиляція. У цьому режимі кондиціонер здійснює тільки вентиляцію повітря і не змінює температуру в кімнаті.

Кондиціонер досить потужний, тому при налаштуванні на правильний режим роботи він за годину доводить температуру в кімнаті до бажаної.

Потрібно написати програму, яка по заданій температурі в кімнаті t_{room}, установленими на кондиціонері бажаної температурі t_{cond} і режиму роботи визначає температуру, яка встановиться в кімнаті через годину.

Формат вхідних даних: Перший рядок вхідного файлу містить два цілих числа t_{room}, і t_{cond}, розділених рівно одним пропуском $(-50 \le t_{room} \le 50 , -50 \le t_{cond} \le 50)$. Другий рядок містить одне слово, записане малими літерами латинського алфавіту - режим роботи кондиціонера.

Формат вихідних даних: Вихідний файл повинен містити одне ціле число - температуру, яка встановиться в кімнаті через годину.

Пояснення до прикладів:

У першому прикладі кондиціонер знаходиться в режимі нагріву. Через годину він нагріє кімнату до бажаної температури 20 градусів.

У другому прикладі кондиціонер знаходиться в режимі охолодження. Оскільки температура в кімнаті нижча, ніж бажана, кондиціонер самостійно вимикається і температура в кімнаті не поміняється.

Приклади вхідних та вихідних даних:

cond.in	cond.out
10 20 heat	20
10 20 freeze	10

Ідея розв'язку задачі Таішева Фердинанда, учня 11 класу Радянськослобідської ЗОШ І-ІІІ ст. Чернігівського району (Трейтяк О.В., учитель інформатики вищої категорії)

Програма на мові Pascal. Використовувався компілятор FreePascal. Програма моделює роботу кондиціонера.

Розв'язок:

```
var a,b: integer;
var c:string;
var t1,t2 :text;
begin
```

```
assign(t1,'cond.in');
assign(t2,'cond.out');
reset(t1);
rewrite(t2);
read(t1,a);
readln(t1,b);
read(t1,c);
if (c='freeze') and (b<a) then writeln(t2,b);
if (c='freeze') and (b>a) then writeln(t2,a);
if (c='heat') and (b>a) then writeln(t2,b);
if (c='heat') and (b<a) then writeln(t2,a);
if (c='fan') then writeln(t2,a);
if (c='auto') then writeln(t2,b);
close(t1);
close(t2);
end.
```

Програма неправильно виконує 3 і 8 тести. Результат - 80 балів із 100.

Ідея розв'язку задачі Харченка В.М., старшого викладача кафедри прикладної математики, інформатики й освітніх вимірювань НДУ імені М. Гоголя, учителя інформатики Ніжинського обласного педагогічного ліцею

Розв'язок:

Зауважимо, що при кожному режимі роботи кондиціонер реалізує деяку функцію, яка обчислює результат за двома аргументами t_{room} і t_{cond}.

У режимі "freeze" кондиціонер реалізує функцію min (t_{room}, t_{cond}), в режим "heat" – функцію $max(t_{room}, t_{cond})$, в режимі "auto" – функцію $f(t_{room}, t_{cond}) = t_{cond}$ (повертає

другий аргумент), а в режимі "fan" – функцію $g(t_{room}, t_{cond}) = t_{room}$ (повертає перший аргумент).

Наведемо програму на мові Паскаль, що реалізує дану ідею:

```
var t1,t2:integer;
    s:string;
begin
assign(input,'cond.in');
reset(input);
readln(t1,t2);
readln(s);
assign(output,'cond.out');
rewrite(output);
ifs='fan'thenwriteln(t1);
ifs='auto'thenwriteln(t2);
ifs='freeze'then
if t1>t2 thenwriteln(t2) elsewriteln(t1);
ifs='heat'then
if t1<t2 thenwriteln(t2) elsewriteln(t1);
end.
```

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Алгоритм на вміння застосовувати розгалуження.

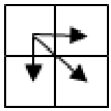
Розв'язок:

```
 var troom,tcond,t:integer; reg:string; f:text;
Begin
 Assign(f,'cond.in');Reset(f);
 Read(f,troom);ReadLN(f,tcond);Readln(f,reg);
```

```
 Close(f);
 If reg='freeze' Then {режим охолодження до бажаної температури}
   If troom<=tcond Then t:=troom Else t:=tcond;
 If reg='heat' Then {режим нагрівання до бажаної температури}
   If troom>=tcond Then t:=troom Else t:=tcond;
 If reg='auto' Then t:=tcond; {автоматична зміна температури до бажаної}
 If reg='fan' Then t:=troom; {режим вентиляції, температури не змінюється}
 Assign(f,'cond.out');Rewrite(f);WriteLN(f,t);Close(f);
End.
```

B - "Поле чудес"

Input file name: wonderland.in
Output file name: wonderland.out
Time limit: 100 ms
Memory limit: 256 M



Степан пройшов до суперфіналу новорічного шоу "Поле чудес". На відміну від звичайних ігор новорічна відрізняється тим, що проводиться не за круглим барабаном, а на прямокутному полі, розбитому на квадратики. Кожен такий квадратик містить одне число (числа можуть бути як доданими, так і від'ємними). Гравець розташовується у верхній лівій клітинці і може переміщуватись у три сусідні клітинки: праву, нижню та праву нижню по діагоналі (див. малюнок) і повинен потрапити до правої нижньої клітинки.

Звісно, що Степан має бажання виграти гру і набрати якомога більше балів. Якщо набрана сума додатна, то Степан виграв, інакше програв.

Напишіть програму, яка допоможе Степану визначити: програє чи виграє він у грі та яку суму він зможе набрати.

Формат вхідних даних: Перший рядок вхідного файлу містить два цілих числа ***N, M (1 ≤ N ≤ 100, 1 ≤ M ≤ 100)*** - розміри ігрового поля.

Наступні ***N*** рядків містять по ***M*** цілих чисел - значення клітинок ігрового поля, по модулю не більші за 1000.

Формат вихідних даних: Перший рядок вихідного файлу має містити одне слово - ***winner***, якщо Степан виграв гру, або ***loser*** - в іншому випадку.

Другий рядок має містити одне число - набрану суму.

Приклади вхідних та вихідних даних:

wonderland.in	wonderland.out
2 3 2 1 1 -1 2 1	winner 6
2 3 1 -2 -1 -2 1 -2	loser 0

Ідея розв'язку задачі Хрол Н.П., учителя інформатики загальноосвітньої спеціалізованої школи І-ІІІ ст. фізико-математичного профілю № 12 м. Чернігова, старшого вчителя

Розв'язок:

Ця задача є класичною задачею з використанням методу динамічного програмування. Розглянути всі можливі маршрути майже нереально.

Подамо поле для гри у вигляді таблиці NxM (у програмі – масив X), у кожну клітинку якої запишемо кількість балів у відповідному квадратику ігрового поля. За умовою задачі Степан хоче набрати максимальну кількість балів. Очевидно, що для цього він повинен вибирати клітинку з максимальною

кількістю балів як на всьому маршруті, так і на кожній його ділянці.

Будемо шукати найкращі шляхи з верхньої лівої клітинки у всі інші. Для цього створимо таблицю «максимальних сум» (у програмі – масив Y).

Не слід забувати і про граничні значення. Оскільки для всіх клітинок таблиці Y, крім верхнього рядка і лівого стовпчика, існує три шляхи, то потрібно виконати окремі обчислення суми для верхньої і лівої границі. Це легко зробити, оскільки для цих клітин існує тільки один маршрут. Наприклад, для верхнього рядка такий:

Для першого приклада:

2→	3→	4

Для другого приклада:

1→	-1→	-2

Для інших клітинок таблиці у клітинку (i,j) ми можемо потрапити тільки з сусідніх клітинок: зліва (i,j-1), зверху (i-1,j) та по діагоналі (i-1,j-1). Пріорітет надамо клітинці з найбільшою кількістю балів. Отже, рухаючись по рядках (або стовпчиках), будемо заповнювати Y, використовуючи формулу:

Y[i,j]:=X[i,j]+max(max(Y[i,j-1],Y[i-1,j]),Y[i-1,j-1]), де max – функція, що визначає більше з двох значень.

Для першого приклада:

Таблиця X

2	1	1
-1	2	1

Таблиця Y

2	3	4
1	5	6

Для другого прикладу:

Таблиця X

1	-2	-1
-2	1	-2

Таблиця Y

1	-1	-2
-1	2	0

Перевіряємо, якщо значення Y[N,M] додатнє, то виводимо слово – ***winner,*** або ***loser*** - в іншому випадку, а також виводимо саме значення клітинки Y[N,M].

Розв'язок:

```
const size=100;
var X,Y:array[1..size,1..size] of int64;
i,j,n,m:integer;
procedure read_data;
begin
  assign(input,'wonderland.in');
  reset(input);
  readln(n,m);
  for i:=1 to n do
    for j:=1 to m do
    read(X[i,j]);
  close(input);
end;
function max(a,b:int64):int64;
begin
 if a>b then max:=a
        else max:=b;
end;
procedure max_sum;
begin
  Y[1,1]:=X[1,1];
  for i:=2 to n do
```

```
    Y[1,i]:=X[1,i]+Y[1,i-1];
  for i:=2 to m do
    Y[i,1]:=X[i,1]+Y[i-1,1];
  for i:=2 to n do
    for j:=2 to m do
    Y[i,j]:=X[i,j]+max(max(Y[i,j-1],Y[i-1,j]),Y[i-1,j-1]);
end;
procedure write_sol;
begin
  assign(output,'wonderland.out');
  rewrite(output);
  if Y[n,m]>0 then  writeln('winner')
                else  writeln('loser');
  writeln(Y[n,m]);
  close(output);
end;
begin
  read_data;
  max_sum;
  write_sol;
end.
```

Програма неправильно виконує 3 і 12 тести. Результат - 90 тестів із 100.

Ідея розв'язку задачі Дасюка Антона, учня 10 класу спеціалізованої загальноосвітньої школи № 2 I-III ст. з поглибленим вивченням іноземних мов м. Чернігова (Коваленко О.І., учитель-методист, учитель інформатики СЗНЗ № 2 м. Чернігова)

Розв'язок:

```
#include <cstdio>
```

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
const int N=101;

int main(){
  freopen("input.txt","r",stdin);
  freopen("output.txt","w",stdout);

  int a, b, i, j, k = 0, m, n, x[N][N], y[N][N] = {};

  for(i = 1;i < N;++ i) y[0][i] = y[i][0] = -1000000000;
  scanf("%d%d", &n, &m);
  for(i = 1;i <= n;++ i)
    for(j = 1;j <= m;++ j)
        {
           scanf("%d", &x[i][j]);
           y[i][j] = x[i][j] + max(y[i-1][j-1], max(y[i-1][j], y[i][j-1]));
        }
  if(y[n][m] > 0)
        printf("winner\n");
  else
        printf("loser\n");
  printf("%d\n",y[n][m]);
  return 0;
}
```

Ідея розв'язку задачі Харченка В.М., старшого викладача кафедри прикладної математики, інформатики й освітніх вимірювань НДУ імені М. Гоголя, учителя інформатики Ніжинського обласного педагогічного ліцею

Розв'язок:

При розв'язуванні даної задачі неважко помітити, що на кожному кроці гри слід вибирати максимальне із трьох можливих чисел. Враховуючи, що максимальні розміри матриці 100x100, то можна говорити про скінченну кількість підзадач знаходження максимального числа. Тому для розв'язання даної задачі скористаємося методом динамічного програмування.

Будемо в програмі використовувати дві двовимірні матриці – одну для зберігання даних з умови задачі, а іншу для зберігання результатів розв'язання підзадач на кожному із кроків. Оскільки за умовою задачі числа не будуть більшими за 1000, то можна було б для першої матриці використовувати тип integer. При зберіганні результатів виконання підзадач матимемо на увазі, що вказаний напрямок руху дає можливість побувати щонайбільше у 200 полях, а значить максимальне значення буде в межах типу LongInt.

Зрозуміло, що комірка b[1,1] матиме те ж значення, що й a[1,1]. Рухаючись по квадратах першого рядка і першого стовпця, легко помітити, що біжуча сума залежить відзначення попередньої комірки і значення числа в даному квадраті. В усіх інших випадках, знаходячи суму чисел пройдених квадратиків, слід знайти три числа b[i-1,j-1]+a[i,j] (здійснюється рух по діагоналі),b[i-1,j]+a[i,j] (рух вниз) та b[i,j-1]+a[i,j] (рух праворуч), і вибрати максимальне з них.

Наведемо програму на мові Паскаль, що реалізує дану ідею:

```
var n,m,i,j:integer;
```

```
a:array [1..100,1..100] of integer;
b:array [1..100,1..100] of longint;
function max(x,y,z:longint):longint;
//функція знаходження максимального серед 3-х цілих чисел
begin
max:=x;
if max<y then max:=y;
if max<z then max:=z;
end;
procedure readdata;
// процедура зчитування інформації з файла
begin
assign(input,'wonderland.in');
reset(input);
readln(n,m);
for i:=1 to n do
for j:=1 to m do read(a[i,j]);
close(input);
end;
procedure run;
//процедура розв’язання підзадач методом динамічного програмування
begin
b[1,1]:=a[1,1];
for i:=2 to m do b[1,i]:=b[1,i-1]+a[1,i];
 for j:=2 to n do  b[j,1]:=b[j-1,1]+a[j,1];
for i:=2 to n do
 for j:=2 to m do b[i,j]:=max(b[i-1,j-1]+a[i,j],b[i-1,j]+a[i,j],b[i,j-1]+a[i,j]);
end;
procedure writedata;
//процедура запису відповіді у файл
begin
assign(output,'wonderland.out');
rewrite(output);
```

```
if b[n,m]>0 then writeln('winner')
else writeln('loser');
writeln(b[n,m]);
close(output);
end;
begin
//обнулення значень елементів масиву, у якому зберігатимуться
//результати розв'язання підзадач
fillchar(b,sizeof(b),0);
readdata;
run;
writedata;
end.
```

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики, ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Розв'язок задачі може бути реалізований з використанням відомого алгоритму знаходження шляху від клітинки (1;1) до клітинки (n;m) з найбільшою сумою.

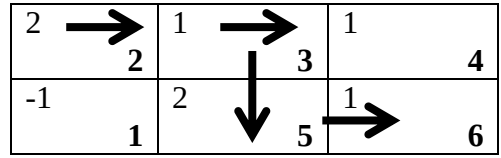
Розглянемо для першого прикладу таблицю набраних балів для кожної клітинки. У першому рядку і стовпчику набрані бали легко знайти, оскільки шлях руху єдиний.

2 → **2**	1 → **3**	1 → **4**
-1 ↓ **1**	2	1

У клітинку (2;2) можна потрапити з трьох клітинок (1;1), (1;2) та (2;1). Вибирати потрібно клітинку з найбільшою кількістю балів. Очевидно, що це клітинка (1;2).

2 → **2**	1 → **3**	1 → **4**
-1 ↓ **1**	2 ↓ **5**	1

Такі ж міркування потрібно провести і для решти клітинок таблиці. Тоді в останній клітинці таблиці буде записана максимально можлива сума.




```
 var i,j,m,n,x,y:integer; s:int64;
    a:array[1..100,1..100] of int64; f:text;
Begin
 Assign(f,'wonderland.in');Reset(f);
 Read(f,n);Read(f,m);
 For i:=1 to n do For j:=1 to m do Read(f,a[i,j]);{зачитування
даних}
 Close(f);
 s:=0;
 For j:=1 to m do Begin s:=s+a[1,j];a[1,j]:=s;End; {формування сум
для першого рядка таблиці}
 s:=0;
 For i:=1 to n do Begin s:=s+a[i,1];a[i,1]:=s;End; {формування сум
для першого стовпчика}
 If (i>1) and (j>1) then
  For i:=2 to n do {розгляд решти клітинок таблиці}
  Begin
   For j:=2 to m do
   Begin
    x:=i-1; y:=j-1;
    If a[x,y]<a[i,j-1] Then x:=i;{визначення оптимального
шляху…}
    If a[x,y]<a[i-1,j] Then Begin y:=j;x:=i-1; End;{…для даної
клітинки }
    a[i,j]:=a[i,j]+a[x,y]; {запис найбільшої суми для даної
клітинки}
   End;
  End;
```

```
 Assign(f,'wonderland.out');Rewrite(f);
 If a[n,m]>0 Then WriteLn(f,'winner',a[n,m])
          Else WriteLn(f,'loser',a[n,m]);
 Close(f);
End.
```

C. - Winter

Input file name: winter.in
Output file name: winter.out
Time limit: 500 ms
Memory limit: 256 M

Країна Ужляндія славиться своїми ідеальними дорогами, але навіть вони не витримали цьогорічної аномально холодної та сніжної зими. Деякі з доріг виявилися заблокованими для руху автомобілістів. Внаслідок цього порушився зв'язок між містами Ужляндії. Два міста країни вважаються з'єднаними, якщо можна дістатися з одного міста в інше, рухаючись не заблокованими дорогами, можливо, через інші міста.

Сусідня братська держава відома на весь світ своїми унікальними обігрівачами. Керівництво країни вирішило надати Ужляндії гуманітарну допомогу. Було вирішено, що обігрівачі доставлятимуться на гвинтокрилі, а далі за допомогою вантажівок розвозитимуться по містах. Оскільки авіаційне паливо не дешеве, потрібно мінімізувати кількість приземлень гвинтокрила так, щоб кожне місто отримало необхідні обігрівачі. Будь ласка, якомога швидше порахуйте цю кількість і врятуйте мешканців Ужляндії.

Формат вхідних даних: В першому рядку записано два числа N і M $(1 \le N \le 100000, 0 \le M \le 200000)$ – кількість міст в Ужляндії та кількість не заблокованих доріг відповідно. В наступних M рядках записано по два числа i та j $(1 \le i, j \le N)$, що значить дорога між містами з номерами i та j не заблокована. Міста в Ужляндії нумеруються в 1 до N.

Формат вихідних даних: В єдиному рядку виведіть мінімальну кількість приземлень гвинтокрила.

Пояснення до прикладу:

Міста 1, 2 та 3 з'єднані між собою, а тому щоб забезпечити їх обігрівачами, необхідно здійснити одне приземлення в одному з цих міст, далі обігрівачі доставлять вантажівками. Міста 4 та 5 зв'язані між собою, тому треба ще одне приземлення. І нарешті місто 6, яке ізольоване від інших, щоб доставити обігрівачі в це місто, треба окреме приземлення гвинтокрила. Всього виходить 3 приземлення.

Приклади вхідних та вихідних даних:

winter.in	winter.out
6 4 3 1 1 2 5 4 2 3	3

Ідея розв'язку задачі Харченка В.М., старшого викладача кафедри прикладної математики, інформатики й освітніх вимірювань НДУ імені М. Гоголя, учителя інформатики Ніжинського обласного педагогічного ліцею

Розв'язок:

За умовою задачі маємо N міст, деякі з яких з'єднані між собою дорогами. Очевидно, що доставити вантажівками обігрівачі можна лише в ті міста, які зв'язані дорогами. У таку групу міст потрібно лише одне приземлення гвинтокрила. Необхідно визначити кількість таких груп, які і будуть кількістю приземлень гвинтокрила.

Сформулюємо математичну модель даної задачі. Дано неорієнтований граф із N вершинами та M ребрами, які задано списком. Підрахувати кількість компонент зв'язності.

Підрахувати кількість компонент зв'язності в неорієнтованому графі, що складається з N вершин та M ребер, можна за допомогою алгоритму пошуку вглиб. Оскільки кількість вершин N≤ 100000, то використовувати рекурсивний алгоритм не можна. Зауважимо, що застосувати матрицю суміжності також не вдасться через установлене обмеження на використовувану пам'ять і велику кількість вершин. Тому доведеться послуговуватися списком суміжності. Він є структурою даних, яка для кожної вершини графа зберігає список суміжних з нею вершин. Список є масивом вказівників, і-ий елемент якого містить покажчик на список вершин, суміжних з і-ою вершиною.

При створенні таких списків вважається, що граф є орієнтованим, тому задіємо процедуру додавання ребер Add двічі: для (u, v)і (v, u). Масив used набуває значення true, якщо у вершині вже були, і false – якщо у ній ще не були.

Складність такого алгоритму O(N+M). Отриманий результат задовольняє умову оптимальності, а тому буде відповіддю на задачу.

Наведемо програму на мові Паскаль, що реалізує дану ідею:

```
Var n, m, u, v, kil, i:LongInt;
    used:array [1..100000] of boolean;
    head: array [1..100000] of LongInt;
    // індекси початків списків
    next: array [1..400002] of LongInt;
    // індекси наступних елементів списків
    value: array [1..400002] of LongInt;
    // значення елементів списків
    free_pos: LongInt;
    // наступна вільна позиція, спочатку встановити 1.
```

```
Procedure Add(v, u: LongInt);
//процедура додавання ребра (v->u) у список суміжності
begin
        next[free_pos] := head[v];
// у вільну комірку наступним елементом робимо початок списку
вершини V
        value[free_pos] := u;
// у цю комірку записуємо інформацію про ребро (кінець ребра)
        head[v] := free_pos;
//установлюємо новий початок списку вершин V
        Inc(free_pos);
//збільшуємо значення вільної позиції.
end;
procedure Dfs(v: LongInt);
//пошук вглиб починаючи з вершини v->
Var u, i: LongInt;
//допоміжні змінні, де i - індекс у списках, u - кінець ребра
begin
        used[v] := true;
        i := head[v];
//встановлюємо на початок списку
        while i <> 0 do
//поки не покаже на кінець списку
        begin
                u := value[i];
                if (not used[u]) then
                        Dfs(u);
                i := next[i];
        end;
end;
procedure initial;
// ініціалізація масивів
begin
fillchar(used,sizeof(used),false);
fillchar(head,sizeof(head),0);
```

```
fillchar(next,sizeof(next),0);
fillchar(value,sizeof(value),0);
end;
procedure run;
// процедура знаходження компонент зв’язності пошуком вглиб
Begin
//зчитування граничних значень вершин і ребер
assign(input,'winter.in');
reset(input);
readln(n, m);
//установка покажчика на перший елемент списку суміжності
free_pos := 1;
//обнуління кількості компонент зв’язності
kil := 0;
for i := 1 to m do
begin
//зчитування нового ребра
readln(u, v);
// додавання ребер у списки суміжності
Add(u, v);
Add(v, u);
end;
for i:=1 to n do
begin
//якщо вершина не пройдена – задіяти пошук вглиб
if (not used[i]) then
begin
// додати до кількості компонент 1
Inc(kil);
Dfs(i);
end
end;
close(input);
end;
procedure writedata;
```

```
// записування результату у файл
begin
assign(output,'winter.out');
rewrite(output);
writeln(kil);
close(output);
end;
begin
initial;
run;
writedata;
end.
```

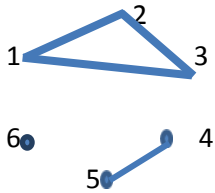
Ідея розв'язку задачі Дасюка Антона, учня 10 класу спеціалізованої загальноосвітньої школи № 2 I-III ст. з поглибленим вивченням іноземних мов м. Чернігова (Коваленко О.І., учитель-методист, учитель інформатики СЗНЗ № 2 м. Чернігова)

Розв'язок:

```
#include <iostream>
#include <vector>

using namespace std;
int n, count = 0;
vector < vector<int>> Graph;
vector<bool> a;

void dfs(int v)
{
 a[v] = 1;
   for(int i = 0; i < Graph[v].size(); i++)
 {
   int to = Graph[v][i];
```

```
  if (!a[to]) dfs(to);
 }

}

int main()
{
int m;
cin >> n >>  m;
Graph.assign(n + 1, vector <int> () );
a.assign (n + 1, 0);
for(int i = 0; i < n; i ++)
  a[i] = 0;

for (int i = 0; i < m; i ++)
{
  int x, y;
   cin >> x >> y;
   Graph[x-1].push_back(y-1);
   Graph[y-1].push_back(x-1);
}

for (int i = 0; i < n; i ++ )
if(!a[i])
  {
    count ++;
    dfs(i);
  }

cout  <<  count <<  endl;

return 0;
}
```

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста



Для розв'язку задачі використано пошук кількості островів незв'язного графа. Островами будемо називати групи міст, що не зв'язані дорогами з іншим містами. Для даних задачі можна утворити такі три острови: {1,2,3},{4, 5}, {6}.

Розв'язок:

```
 var i,j,m,n,x,y,z,k,l:longint; s:int64;
     a:array[1..100000] of longint; f:text;
Begin
 Assign(f,'winter.in');Reset(f);
 s:=1; z:=0; FillChar(a,SizeOf(a),0);
 Read(f,n);Read(f,m);
 For i:=1 to m do
 Begin
  Read(f,x); Read(f,y);
  If (a[x]=0) and (a[y]=0) {якщо міста не належать жодному острову, }
    Then Begin a[x]:=s; a[y]:=s; s:=s+1;End {то створюємо номер нового острову}
    Else  {якщо хоча б одне місто належить якому-небудь острову}
      Begin
```

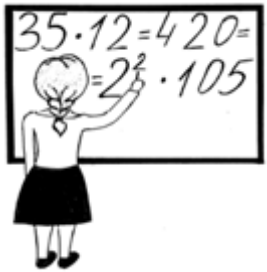
```
    If (a[x]>0) and (a[y]>0) Then {якщо обидва міста належать до якихось островів}
     Begin {якщо міста належать до різних островів, то міста з більшим номером острова "переносяться" до острова з меншим номером}
       If a[x]>a[y]
        Then Begin k:=a[y]; l:=a[x]; z:=z+1;End;
       If a[x]<a[y]
        Then Begin k:=a[x]; l:=a[y]; z:=z+1;End;
       If a[x]<>a[y]
        Then For j:=1 to n do If a[j]=l Then a[j]:=k;
      End;
    If (a[x]>0) and (a[y]=0) Then a[y]:=a[x]; {якщо одне з міст не належить острову}
     If (a[y]>0) and (a[x]=0) Then a[x]:=a[y];
   End;
 End;
 Close(f);
 s:=s-1-z; {формула визначення кількості знайдених островів з кількома містами}
 For i:=1 to n do If a[i]=0 Then s:=s+1; {якщо є міста-острови, то їх також враховуємо}
 Assign(f,'winter.out');Rewrite(f);WriteLN(f,s);Close(f);
End.
```

На сервері програма правильно виконує 20 тестів, 21-25 – обмеження за часом. Результат перевірки на локальному комп'ютері 80 із 100.

D - Гарні числа

Input file name: beautiful.in
Output file name: beautiful.out
Time limit: 1000 ms
Memory limit: 256 M

Школа № 1331 в Ужляндії відома дуже високим рівнем знань своїх учнів з математики, тому що більшість учнів відвідують факультативні заняття відомого вчителя Антона Андрійовича.



Сьогодні Антон Андрійович розказав своїм учням про числа, які, на його думку, можуть володіти рядом цікавих властивостей. Він назвав такі числа гарними. Число називається гарним, якщо не існує такого цілого числа, більшого одиниці, на квадрат якого воно б ділилося без залишку. Наприклад, число 12 не є гарним, тому що воно ділиться на 4, тобто на квадрат числа 2. Числа 13 і 14 є гарними числами.

Учні Антона Андрійовича дуже гарні в усному рахунку, тому в першому завданні необхідно було визначити, чи є деяке число гарним.

Однак, Марися, краща його учениця, швидко впоралась із цим завданням. Щоб якось її зайняти, учитель написав на дошці ***N*** чисел і дав їй нове завдання: визначити, чи є добуток цих чисел гарним числом. Дуже скоро Марися отримала відповідь, однак вона хоче перевірити себе. Тому вона просить Вас написати програму, яка перевіряє: чи є добуток чисел гарним числом, а якщо ні, то їй потрібно знати яке-небудь число, відмінне від одиниці, на квадрат якого ділиться добуток цих чисел.

Формат вхідних даних: Перший рядок вхідного файлу містить число ***N (1 ≤ N ≤ 100)*** - кількість чисел, які вчитель написав на дошці для Марисі. Другий рядок

містить N натуральних чисел - самі числа. Кожне з чисел не перевершує 10^{18}.

Формат вихідних даних: Якщо число є гарним, виведіть єдиний рядок, що складається з слова Beautiful. Інакше, виведіть яке-небудь число, відмінне від одиниці, на квадрат якого ділиться добуток N чисел.

Пояснення до прикладів:

Перший приклад: 5*6*7 = 210. Не існує числа, більшого за одиницю, на квадрат якого 210 ділилось би без залишку, тому 210 - гарне число.

Другий приклад: 35*12 = 420. 420 ділиться на 4, тобто на квадрат числа 2.

Оцінювання:

$A_1 * A_2 * ... * A_N \le 10^6$ – неменше 20 балів

$A_1 * A_2 * ... * A_N \le 10^{12}$ – не менше 40 балів

Для усіх i: $A_i \le 10^{12}$ – не менше 60 балів

Приклади вхідних та вихідних даних:

beautiful.in	beautiful.out
3 5 6 7	Beautiful
2 35 12	2

Ідея розв'язку задачі Хрол Н.П., учителя інформатики загальноосвітньої спеціалізованої школи І-ІІІ ст. фізико-математичного профілю № 12 м. Чернігова, старшого вчителя

Ідея розв'язання:

Якщо добуток чисел ділиться на квадрат цілого числа (наприклад k), то можливі два випадки:

1. Число k є дільником не менше двох чисел з цього добутку.

2. Квадрат числа k є дільником одного з чисел добутку.

У першому випадку будемо шукати НСД усіх попарно чисел масиву А.

У другому випадку будемо шукати квадрат числа, що є дільником одного з чисел.

Розв'язок:

```
var    A:array [1..100] of comp;
       n,i,j:integer;
       k:int64;
function nsd(x,y:comp):comp;
begin
  repeat
  if x>y then  begin
               x:=trunc(x-trunc(x/y)*y);
               end
         else  begin
               y:=trunc(y-trunc(y/x)*x);
               end;
  until (x=0) or (y=0);
  nsd:=x+y;
end;
begin
  assign(input,'beautiful.in');
  assign(output,'beautiful.out');
  reset(input);
  rewrite(output);
  readln(n);
  for i:=1 to n do
  read(A[i]);
  for i:=1 to n-1 do
    for j:=i+1 to n do
    if   trunc(nsd(A[i],A[j]))<>1 then  begin
                                write(trunc(nsd(A[i],A[j])));
                                close(input);
                                close(output);
```

```
                                exit
                                end;
  for i:=1 to n do
    begin
    k:=trunc(sqrt(A[i]));
    if A[i]>1000000000000 then
    while k>trunc(sqrt(A[i]))-1000 do
      begin
      if  (trunc(A[i]-trunc(A[i]/sqr(k))*sqr(k))=0)  then  begin
                                              write(k);
                                              close(input);
                                              close(output);
                                              exit
                                              end;
      k:=k-1;
      end
                        else
    while k>1 do
      begin
      if  (trunc(A[i]-trunc(A[i]/sqr(k))*sqr(k))=0)  then  begin
                                              write(k);
                                              close(input);
                                              close(output);
                                              exit
                                              end;
      k:=k-1;
      end
    end;
  writeln('Beautiful');
  close(input);
  close(output);
end.
```

Ідея розв'язку задачі Дасюка Антона, учня 10 класу спеціалізованої загальноосвітньої школи № 2 І-ІІІ ст. з поглибленим вивченням іноземних мов м. Чернігова (Коваленко О.І, учитель-методист, учитель інформатики СЗНЗ № 2 м. Чернігова)

Розглянемо випадки, коли добуток N заданих чисел ($A_1, A_2, \dots, A_n$) ділитиметься без залишку на квадрат цілого числа, що більше одиниці:

1) якщо $НСД(A_i, A_j) = x > 1$ ($i \neq j$, НСД – найбільший спільний дільник двох чисел (Алгоритм Евкліда знаходження НСД)). Тобто A_i та A_j діляться на x без залишку, тоді $A_i \cdot A_j$ ділитиметься на x^2 без остачі. Отже, x є відповіддю на задачу.
2) якщо A_i - точний квадрат, тоді $\sqrt{A_i}$ – відповідь.
3) якщо $A_i = x^2 \cdot y$, тоді x - відповідь на задачу. Щоби знайти таке x, достатньо перебрати всі дільники числа A_i до кубічного кореня з цього числа ($A_i \vdots z, 1 < z \leq \sqrt[3]{A_i}$)і перевірити наступні умови:
   а) якщо $A_i \vdots z^2$, то z – відповідь;
   б) якщо $\frac{A_i}{z}$ - точний квадрат, то $\sqrt{\frac{A_i}{z}}$ - відповідь.

Якщо жодна з умов не виконується, потрібно вивести "Beautiful".

Розв'язок:

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cmath>
```

```
using namespace std;

int main()
{
 freopen("input.txt", "r", stdin);
 freopen("output.txt", "w", stdout);
 long long a, b, r = 1, z[100];
 int i , j, n;
 cin >> n;
 for(i = 0;i < n;++ i)cin >> z[i];
 for(i = 0;i < n - 1;++ i)// Якщо НОД(z[i],z[j])>1
   for(j = i + 1;j < n;++ j)
    {
     a = z[i],b = z[j];
     while(b > 0)
     {
       a %= b;
       swap(a, b);
     }
     if(a > 1)
        {
           cout << a << endl;
           return 0;
        }
    }

 for(i = 0;i < n;++ i)
    {
       b = (long long) sqrt((long double) z[i]);// Якщо z[i]-
повнийквадрат
       if(b * b == z[i] && b > 1)
          {
             cout << b << endl;
             return 0;
          }
       for(a = 2;a * a * a <= z[i];++ a)
```

```
if(z[i] % a == 0)// Якщо z[i] ділиться на a*a
if(z[i] % (a * a) == 0)
                {
                   cout << a << endl;
                   return 0;
                }
      else
         {// Якщо z[i]/a – повний квадрат
b = (long long) sqrt((long double) (z[i] / a));
            if(a * b * b == z[i] && b > 1)
                {
                   cout << b << endl;
                   return 0;
                }
         }
   }

 cout << "Beautiful\n";

 return 0;
}
```

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики, учителя-методиста ЗОШ І-ІІІ ст. № 7 м. Прилук

Звичайно, задачу можна спробувати розв'язати напряму – отримати добуток чисел і перевіряти подільність на квадрати натуральних чисел. Такий розв'язок має право на існування і принесе певну кількість балів.

Розіб'ємо алгоритм на три частини.

1. Розглядаємо всі можливі пари даних чисел. Якщо НСД якоїсь пари не рівний 1, то добуток точно

ділиться на квадрат знайденого числа. Треба вивести знайдене число.

2. Якщо всі дані числа взаємно прості (тобто перший пункт не виконався), то перевіряємо, чи не буде якесь із чисел квадратом натурального числа.
3. Якщо ж і 2-й пункт не дав результату, то перевіряємо дані числа на подільність на квадрати всіх можливих натуральних чисел, що менші за дані числа.
4. У разі неспрацьовування пункту 3 залишається констатувати "гарність" числа.

```
 var i,y:longint; j,n,x,z:int64; a:array[1..100] of int64;
   f:text; ok:boolean;

Function GCD( a,b:int64 ):int64; {рекурсивна функція знаходження НСД двох чисел}
 Begin
  If b=0 Then GCD:=a Else GCD:=GCD(b,a mod b);
 end;

Begin
 Assign(f,'beautiful.in');Reset(f);
 Read(f,n);
 For i:=1 to n do Read(f,a[i]);Close(f);
{1}
 x:=1;
 For i:=1 to n-1 do
 Begin
  y:=i+1;
  While (y<=n) and (x=1) do
  Begin
   If GCD(a[i],a[y])>x Then x:=GCD(a[i],a[y]);
   y:=y+1;
  End;
  If x>1 Then Break;
 End;
```

```
{2}
 If x=1 Then
 Begin
  For i:=1 to n do
  Begin
    j:=Trunc(sqrt(a[i]));
    If Sqr(j)=a[i] Then x:=j;
    If x>1 Then Break;
  End;
 End;
{3}
 If x=1 Then
 Begin
  y:=1;
  While (y<=n) and (x=1) do
  Begin
    z:=1;
    While (sqr(Trunc(sqrt(a[y])))<>a[y]) and (y<=n) do
    Begin
     If z<=2 Then z:=z+1
Else Begin z:=z+2; If z mod 3=0 Then z:=z+2; End;
     If a[y] mod z=0 Then a[y]:=a[y] div z;
    End;
    If sqr(Trunc(sqrt(a[y])))=a[y] Then x:=Trunc(sqrt(a[y]));
    If x>1 Then Break;
    y:=y+1;
  End;
 End;
{4}
 Assign(f,'beautiful.out');Rewrite(f);
 If x>1 then WriteLN(f,x) else WriteLN(f,'Beautiful');
 Close(f);
End.
```

E - Вправи Степана

Input file name:	exercises.in
Output file name:	exercises.out
Time limit:	1000 ms
Memory limit:	128 M

Степан вирішив досягти успіху не тільки в спортивному програмуванні, а й у спорті. На жаль, пройшло вже багато часу з дня його останнього тренування, тому, щоб набрати хорошу форму, доведеться починати з нуля.

Придумати вправи для тренувань виявилося непросто. Тому Степан вирішив пошукати ідеї в Інтернеті. Він знайшов сайт, на якому пропонувалася кілька серій тренувальних вправ. Кожна серія тренувань за планом займає N днів. У кожен із цих N днів пропонується робити одну «вправу дня», а також до нього додаються рекомендації щодо виконання у вигляді $"A_i - B_i"$. Це позначає, що для підвищення рівня сили потрібно виконувати вправу від A_i до B_i раз. Якщо виконувати вправу менш, ніж A_i, або більш, ніж B_i раз, то це принесе скоріш шкоду, ніж користь. Степан не хоче завдавати собі шкоди, тому буде робити від A_i до B_i раз, або зовсім не робити цю вправу.

Почитавши всі описи вправ, Степан зрозумів, що цей курс не розрахований на новачків, але вирішив не здаватися і адаптувати курс вправ під себе. Він знає, що при вивченні i-ї вправи йому доведеться втратити K_i рівнів сили, зате за виконання вправи X раз його рівень сили зросте на F_i*X. Степан не може вивчити і виконати вправу, якщо його поточний рівень сили менший за K_i. У дні, коли Степану не вистачає сил або часу тренуватись, він може пропускати тренування, і рівень його сили залишається без зміни. Знаючи свої можливості, Степан розуміє, якщо в якийсь день він виконає вправу більш T разів, то наступні D днів він буде занадто втомленим, щоб займатися.

Якщо Степан виконає вправу більш T разів в якийсь з останніх T днів серії тренувань, то він почне відпочивати з наступного дня, а закінчить вже після кінця серії. Степан хоче

отримати від занять максимум користі, тому він планує витратити на них ***N*** днів! Для кожної серії тренувань допоможіть йому визначити максимальний рівень сили, який він зможе досягти в кінці тренувань. До початку тренувань рівень сили Афанасія дорівнює нулю.

Формат вхідних даних: Перший рядок вхідного файлу містить єдине ціле число N $(1 \le N \le 10^5)$ - кількість днів тренувань.

Другий рядок містить два цілих числа T, D $(1 \le T \le 10^6, 1 \le D \le 10^5)$, якщо в якийсь день Степан виконає вправу більш T разів, то йому доведеться відпочивати D наступних днів. Наступні N рядків описують вправи, $i+2$-ий рядок містить опис вправи в день i. Кожна вправа описується числами A_i, B_i, K_i, F_i $(0 \le K_i \le 10^9, 1 \le A_i \le B_i \le 10^6, 1 \le F_i \le 10^6)$, розділеними одиночними пробілами, - де A_i, B_i відповідно рекомендований мінімум і максимум кількості разів виконання вправи, K_i- кількість втрачаються рівнів сили при вивченні вправи, F_i- кількість рівнів сили, одержуваних за кожен раз виконання вправи.

Формат вихідних даних: Перший рядок вихідного файлу має містити одне ціле число S - максимальний рівень сили, який Степан може досягти до кінця тренувань. Наступний рядок повинен містити N цілих чисел X_i - кількість разів виконання вправи в день i, якщо в i-ий день він відпочивав, то виведіть 0.

Пояснення до прикладів:

Щоб досягти максимального рівня сили, треба в перший день виконувати вправу 4 рази, щоб не довелося пропускати наступний день. У другий день Афанасій зможе збільшити свій рівень ще на 790 (втрачає 10 рівнів при вивченні та виконує вправу 8 разів), але тоді він не зможе займатися 1 день (третій день).

У четвертий день він збільшує свій рівень на 48, так як Степан виконав вправу більше 4 разів, він змушений пропустити п'ятий.

Приклади вхідних та вихідних даних:

exercises.in	exercises.out
5 4 1 3 5 0 10 6 8 10 100 2 8 10 15 5 6 0 8 2 2 1 7	878 4 8 0 6 0

Ідея розв'язку задачі Дасюка Антона, учня 10 класу спеціалізованої загальноосвітньої школи № 2 I-III ст. з поглибленим вивченням іноземних мов м. Чернігова (Коваленко О.І., учитель-методист, учитель інформатики СЗНЗ № 2 м. Чернігова)

За умовою задачі є N днів, у кожен з яких можна виконувати або не виконувати задану вправу. Вправу можна виконувати від A_i до B_i разів, та якщо рівень сили перед її виконанням не менше K_i. Якщо виконати вправу X разів, то рівень сили зросте на $F_i \cdot X$, при цьому якщо $X > T$, то наступні D днів потрібно буде відпочивати.

Для розв'язання даної задачі скористаємося методом динамічного програмування. Потрібно помітити, що для оптимального накопичення сили, вправу потрібно виконувати або A_i, або B_i, або T разів, інша кількість не покращить результат. Зберігатимемо динаміку V_i - максимальний рівень сили, якого можна досягти в день i. Отже матимемо наступні переходи:

1) $V_{i+1} = max(V_{i+1}, V_i)$, тобто і-го дня вправа не виконується;

2) $V_{i+1} = \max(V_{i+1}, V_i - K_i + F_i \cdot T)$, якщо і-го дня Степан виконає вправу T разів, при чому $A_i \le T \le B_i$;
3) $V_{i+1} = \max(V_{i+1}, V_i - K_i + F_i \cdot B_i)$, якщо і-го дня Степан виконає вправу B_i разів, при чому $B_i \le T$;
4) $V_{i+D+1} = \max(V_{i+D+1}, V_i - K_i + F_i \cdot B_i)$, якщо і-го дня Степан виконає вправу B_i разів, при чому $B_i > T$;

Звичайно, перехід відбуватиметься, якщо $V_i \ge K_i$, для всіх чотирьох випадків. Також потрібно зберігати додатковий масив, для відновлення шляху. Безпосередньо відповідь буде дорівнювати максимальному елементу масиву V.

Розв'язок:

```
#include <cstdio>
#include <iostream>
#include <algorithm>

using namespace std;

const int N = 100001;

int main(){

 freopen("input.txt","r",stdin);
 freopen("output.txt","w",stdout);
 int a, b, d, f, i, k, m, n, t, z[N] = {};
 long long v[N] = {},s = 0,u = 1;
 pair <int, int> P[N];

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

 for(i = 0;i < n;++ i)
   {
      scanf("%d%d%d%d",&a, &b, &k, &f);
      if(v[i + 1] <= v[i])
         {
            v[i + 1] = v[i];
```

```
            P[i + 1].first = i;
            P[i + 1].second = 0;
        }
    if(v[i] >= k)
        {
            if(b <= t)
             {
                if(v[i] - k + u * f * b > v[i+1])
                    {
                        v[i + 1] = v[i] - k + u * f * b;
                        P[i + 1].first = i;
                        P[i + 1].second = b;
                    }
             }
            else
             {
                if(t >= a && v[i] - k + u * f * t > v[i+1])
                    {
                        v[i + 1] = v[i] - k + u * f * t;
                        P[i + 1].first = i;
                        P[i + 1].second = t;
                    }
                m = min(n, i + d + 1);
                if(v[i] - k + u * f * b > v[m])
                    {
                        v[m]=v[i] - k + u *f * b;
                        P[m].first = i;
                        P[m].second = b;
                    }
            }
        }
    if(v[i] > s)
        s = v[i];
}
```

```
if(v[n] > s)
   s = v[n];
cout << s << endl;
i = n;

while(i)
   {
      a = P[i].first;
      z[a] = P[i].second;
      i = a;
   }
printf("%d", z[0]);
for(i = 1;i < n;++ i)
   printf(" %d", z[i]);
cout << endl;

return 0;
}
```

Ідея розв'язку задачі Харченка В.М., старшого викладача кафедри прикладної математики, інформатики й освітніх вимірювань НДУ імені М.Гоголя, учителя інформатики Ніжинського обласного педагогічного ліцею

Розв'язок:

За умовою задачі є N днів, у кожен з яких можна виконувати або не виконувати задану вправу. Вправу можна виконувати від A_i до B_i разів, якщо рівень сили перед її виконанням не менше K_i. Якщо виконати вправу X разів, то рівень сили зросте на $F_i \cdot X$, при цьому якщо $X > T$, то наступні D днів потрібно буде відпочивати.

Для розв'язання даної задачі скористаємося методом динамічного програмування. Легко помітити, що для оптимального накопичення сили, вправу потрібно виконувати або A_i, або B_i, або T разів, інша кількість не покращить результат. Зберігатимемо динаміку mx_i – максимальний рівень сили, якого можна досягти в день i. Отже матимемо наступні переходи:

1. $mx_{i+1} = max(mx_i, mx_{i+1})$, тобто і-го дня вправа не виконується;
2. $mx_{i+1} = \max(mx_i, mx_i - K_i + F_i \cdot T)$, якщо і-го дня Степан виконає вправу T разів, при чому $A_i \le T \le B_i$;
3. $mx_i = \max(mx_i, mx_i - K_i + F_i \cdot B_i)$, якщо і-го дня Степан виконає вправу B_i разів, при чому $B_i \le T$;
4. $mx_{i+D+1} = \max(mx_{i+D+1}, mx_i - K_i + F_i \cdot B_i)$, якщо і-го дня Степан виконає вправу B_i разів, при чому $B_i > T$;

Звичайно, перехід відбуватиметься, якщо $mx_i \ge K_i$, для всіх чотирьох випадків. Також потрібно зберігати додатковий масив, для відновлення шляху. Безпосередньо відповідь буде дорівнювати максимальному елементу масиву mx_{n-1}.

Наведемо програму на мові Паскаль, що реалізує дану ідею:

```
usessysutils;
var n, t, d: longint;
   i, day, cur, prev: longint;
   a, b, k, f, days: array[0..100000] of int64;
//масиви, у яких зберігатимуться зчитуванні з файла дані
//days – масив для зберігання кількості вправ, виконаних Степаном
mx: array[0..100000+1] of int64;
// масив для зберігання накопиченої Степаном сили
lst, done: array[0..100000+1] of int64;
```

```
// масива списку днів та допоміжного масива кількості виконаних вправ
have, tomorrow: int64;
//наявні на даний та вчорашній день сили
functionmin(x,y:int64):int64;
//функція знаходження мінімального значення серед 2-х цілих чисел
begin
min:=x;
if y<minthenmin:=y;
end;
procedure initial;
//процедура ініціалізації масивів
begin
fillchar(lst,sizeof(lst),-1);
fillchar(a,sizeof(a),0);
fillchar(b,sizeof(b),0);
fillchar(k,sizeof(k),0);
fillchar(days,sizeof(days),0);
fillchar(done,sizeof(done),0);
end;
procedure readdata;
//процедура зчитування інформації з файла
begin
assign(input, 'exercises.in');
reset(input);
read(n);
read(t, d);
for i := 0 to n - 1 doread(a[i], b[i], k[i], f[i]);
end;
procedure writedata;
//процедура запису відповіді у файл
begin
assign(output, 'exercises.out');
rewrite(output);
```

```
writeln(mx[n]);
for i := 0 to n - 1 do
begin
if (i > 0)  thenwrite(' ');
write(days[i]);
end;
writeln;
end;
procedure run;
begin
forday := 0 to n - 1 dobegin
// якщо після виконання вправ кількість сили зменшується, то вправи
// Степан не виконує
if (mx[day + 1] <mx[day]) then
        begin
        mx[day + 1] := mx[day];
        lst[day + 1] := lst[day];
        done[day + 1] := done[day];
        end;
        have := mx[day];
//якщо сил достатньо для виконання вправ
        if (have>= k[day]) then
        begin
// випадок, коли a[i]<=t<=b[i]
        if (a[day] <= t) and (t <= b[day]) then begin
        tomorrow := have - k[day] + t * f[day];
        if (mx[day + 1] <tomorrow) then begin
                mx[day + 1] := tomorrow;
                lst[day + 1] := day;
                done[day + 1] := t;
        end;
        end;
// випадок, коли b[i] <=t
        if (b[day] <= t) then begin
```

```
        tomorrow := have - k[day] + b[day] * f[day];
        if (mx[day + 1] <tomorrow) then begin
                mx[day + 1] := tomorrow;
                lst[day + 1] := day;
                done[day + 1] := b[day];
        end;
        endelsebegin
//альтернативний випадок з контролем індекса масиву
                tomorrow := have - k[day] + b[day] * f[day];
                if (mx[min(n, day + 1 + d)] <tomorrow) then begin
                mx[min(n, day + 1 + d)] := tomorrow;
                lst[min(n, day + 1 + d)] := day;
                done[min(n, day + 1 + d)] := b[day];
                end;
        end;
        end;
// повернення списку днів на початок та формування значень масиву
// виконання вправ
cur := n;
while (true) do
begin
        prev := lst[cur];
        if (prev< 0) then break;
        days[prev] := done[cur];
        cur := prev;
end;
end;
begin
initial;
readdata;
run;
writedata;
end.
```

8-11 класи
II тур
A - "Усе, Степан! Ти мене дістав!"

Input file name:	bubble.in
Output file name:	bubble.out
Time limit:	100 ms
Memory limit:	256 M

Степан нещодавно відпочивав у Японії і привіз звідти нову жувальну гумку. На першій парі в університеті він поділився гумкою зі своїм товаришем. Дочекавшись моменту, коли лектор повернувся до дошки, на рахунок "три - чотири" хлопці дружньо почали надувати бульбашки. Відомо, що Степан надуває бульбашку до максимально можливого розміру за час t_1, після чого бульбашка миттєво лопається, і Степан починає надувати бульбашку заново з тією ж швидкістю. Товариш Степана робить те ж саме за час t_2.

Увесь цей час викладач настільки захоплений доведенням теореми, що взагалі нічого не чує. І тільки коли обидві бульбашки лопнуть одночасно, викладач почує шум і обернеться. І тоді вже точно студентам попаде на горіхи, а більше усього тому, хто приніс на пару жувальні гумки.

Визначте, скільки часу хлопці можуть насолоджуватись надуванням бульбашок, не замічені викладачем.

Наприклад, якщо $t_1 = 2$, $t_2 = 3$, то буде відбуватись наступне:

Степан надуває бульбашку з моменту часу $t = 0$ до моменту часу $t = 2$, потім бульбашка лопається, і він надуває бульбашку знову - з моменту часу $t = 2$ до моменту часу $t = 4$, а потім ще раз - з моменту часу $t = 4$ до $t = 6$.

Товариш Степана надуває бульбашку з $t = 0$ до $t = 3$ і ще раз з $t = 3$ до $t = 6$.

У момент часу $t = 6$ бульбашки лопаються одночасно в обох студентів, викладач повертається і каже: "Усе, Степан! Ти мене дістав!".

Формат вхідних даних: Перший рядок вхідного файлу містить два цілих числа t_1, t_2 $(1 \le t_1, t_2 \le 10^9)$.

Формат вихідних даних: Вихідний файл повинен містити одне ціле число - час, протягом якого Степан з товаришем можуть насолоджуватись надуванням бульбашок.

Приклад вхідних та вихідних даних:

bubble.in	bubble.out
2 3	6
1 16	16

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Потрібно знайти НСК двох даних чисел.

Розв'язок:

```
Var m,n,mn:int64; f:text;
Begin
 Assign(f,'bubble.in');Reset(f);Read(f,n);Read(f,m);Close(f);
 mn:=m*n;
 While m <> n do If m>n then m:=m-n else n:=n-m; {пошук НСД
двох чисел за алгоритмом Евкліда}
 Assign(f,'bubble.out');Rewrite(f);
 Writeln(f,mn div m); {формула НСК(m,n)=m*n/ НСД(m,n)}
 Close(f);
End.
```

B - Степан – бізнесмен

Input file name:	businessman.in
Output file name:	businessman.out
Time limit:	300 ms
Memory limit:	256 M

Ужляндія, як відомо, країна з розвиненими торговими відносинами.

Степан вирішив спробувати зайнятися торгівлею і підзаробити собі на відпустку продажем комп'ютерної техніки. Для цього йому необхідно закуповувати техніку у інших продавців. Перш ніж почати роботу, він вирішив простежити за подіями на ринку Ужляндії і придумати, як отримувати найбільший прибуток.

Степан дізнався, що кожен продавець продає один комп'ютер, і кожен покупець готовий купити рівно один комп'ютер. Усього на ринку торгують N продавців, вартість комп'ютера у i-го з них дорівнює A_i Ужляндських монет, причому ціни можуть відрізнятися у різних продавців. Крім того, він знайшов для себе M потенційних покупців, кожен з яких хоче купити комп'ютер за B_i монет. При цьому сам Степан може купити і продати будь-яку кількість комп'ютерів.

Степану необхідно отримати найбільший прибуток (вигідно купити і вигідно продати). Тому він звернувся за допомогою до Вас - кращому програмісту Ужляндії.

Формат вхідних даних: Перший рядок вхідного файлу містить розділення одиночним пропуском два цілих числа N, M $(1 \le N, M \le 10^5)$ - кількість продавців на ринку Байтландії і кількість потенційних покупців відповідно. Другий рядок містить N цілих чисел A_i $(0 \le A_i \le 10^9)$ - вартості, за якими продавці готові продавати комп'ютери. Третій рядок містить M цілих чисел B_i $(0 \le B_i \le 10^9)$ - суми, які потенційні покупці готові віддати при покупці комп'ютера.

Формат вихідних даних: Перший рядок вихідного файлу має містити одне ціле число ***S*** - розмір максимальної вигоди в монетах, яку може отримати Степан.

Пояснення до прикладів:

У першому прикладі Степан купує комп'ютери в 1-го і 2-го продавців і продає їх будь-яким двом покупцям. У другому прикладі Степану найбільш вигідно купити комп'ютери у 1-го, 4-го і 6-го продавців і продати 3-му, 4-му і 5-му покупцям.

Оцінювання:

$N + M \le 20$, для усіх i: $A_i, B_j \le 1000$ – не менше 40 балів

$N + M \le 2000$, для усіх i: $A_i, B_j \le 1000$ – не менше 50 балів

$N + M \le 2000$ – не менше 75 балів

Приклад вхідних та вихідних даних:

businessman.in	businessman.out
2 3 1 1 3 3 3	4
6 5 5 10 8 4 7 2 3 1 11 18 9	27

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Для отримання максимального прибутку потрібно купити якомога дешевше, а продати – дорожче. Тож маємо відсортувати дані масиви по зростанню закупівельної ціни та по спаданню ціни продажу. Потім шукаємо прибуток від реалізації кожної партії комп'ютерів. Якщо прибутку немає (недодатній), то така операція не здійснюється.

Розв'язок:

```
type massiv = array[1..100000] of longint;
```

```
 Var  i,j:longint; n,m,r,x:int64;
      a,b:massiv; f:text;l:boolean;
{Сортування швидке}
Procedure QuickSort(var a: massiv; l, r: longint);
 var m, x, y: longint;
Begin
 m := ((l + r) div 2);
 i := l; j := r; x := a[m];
 while i <= j do
 begin
   While a[i] < x do inc(i);
   While a[j] > x do dec(j);
   If i <= j Then
   Begin
     y:=a[i];a[i]:=a[j];a[j]:=y;
     inc(i);
     dec(j);
   End;
 End;
 If l < j Then QuickSort(a, l, j);
 If r > i Then QuickSort(a, i, r);
End;

Begin
 Assign(f,'businessman.in');Reset(f);Read(f,n,m);
 If m>n Then x:=n Else x:=m;
 For i:=1 to n do Read(f,a[i]);
 QuickSort(a, 1, n); {сортування на зростанням}
 For i:=1 to m do Read(f,b[i]);
 QuickSort(b, 1, m); {сортування на зростанням}
 For i:=1 to m div 2 do {пересортування на спаданням}
   Begin r:=b[i]; b[i]:=b[m-i+1];b[m-i+1]:=r;End;
 Close(f);
 i:=1;r:=0;
```

```
While (i<=x) and (b[i]-a[i]>=0) do {якщо різниця >0, то рахуємо прибуток}
  Begin r:=r+b[i]-a[i]; i:=i+1;End;
 Assign(f,'businessman.out');Rewrite(f);WriteLN(f,r);Close(f);
End.
```

C – Transit

Input file name:	transit.in
Output file name:	transit.out
Time limit:	1000 ms
Memory limit:	256 M

Країна Ужляндія має вигідне географічне розташування – її територія знаходиться на перетині важливих торгівельних шляхів. Одним із таких є торгівельний шлях, яким сусідня братська держава доставляє свої унікальні обігрівачі до інших країн.

На кордоні Ужляндії та братської держави, де починається цей шлях, розташований спеціальний пропускний пункт, через який щодня проїжджає потяг із величезною кількістю обігрівачів. Зовсім недавно між урядами двох братських країн було погоджено нові правила транзиту обігрівачів через територію Ужляндії на найближчі N днів. Згідно з новим договором має обратися певне число m – максимальна кількість обігрівачів в одному потязі. Тоді з кожного потяга, що транспортує A_i обігрівачів, буде відвантажено рівно A_i-m одиниць іноземної продукції (звичайно, якщо $A_i > m$, інакше ж потяг рухатиметься без зупинок і нічого відвантажено не буде). Власне це й буде плата за проходження потяга територією Ужляндії, вона еквівалентна грошовим витратам на утримання залізничних колій. Сумарна кількість відвантажених в Ужляндії за N днів обігрівачів повинна бути не менша K, інакше країна зазнає збитків.

Стала відома кількість обігрівачів у потязі в кожен із N днів (ця інформація надається за умовами контракту).

Знайдіть максимальне число ***m***, при якому Ужляндія не зазнає економічних збитків.

Формат вхідних даних: У першому рядку записано два числа ***N, K (1 ≤ N ≤ 10^6, 1 ≤ K ≤ 2*10^9)***. У наступному рядку задано ***N*** чисел – кількість обігрівачів у потязі в кожен з ***N*** днів, що не перевищує 10^9.

Формат вихідних даних: В єдиному рядку виведіть одне число – відповідь на задачу, гарантується, що відповідь завжди існує.

Пояснення до прикладу:

Усього територією Ужляндії пройде 4 потяги з 11, 6, 1 та 8 обігрівачами відповідно. Щоби країна не знала збитків, потрібно відвантажити не менше 7 обігрівачів. Очевидно, що максимальне можливе ***m***, яке задовольнить цю умову, буде рівне 6, тоді з потягів буде відвантажено відповідно 5, 0, 0, 2 обігрівачів, що в сумі дорівнює 7 і задовольняє умову.

Оцінювання:

N ≤ 300 – не менше 20 балів

N ≤ 10000 – не менше 40 балів

Приклад вхідних та вихідних даних:

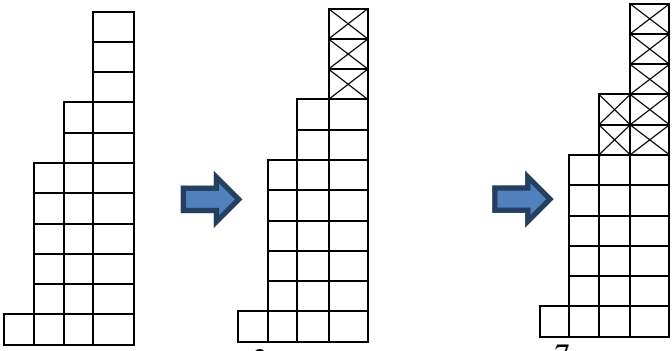
transit.in	transit.out
4 7 11 6 1 8	6

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Розв'язок:

Представимо кількість обігрівачів у потягах у вигляді відсортованої стовпчастої діаграми та на її прикладі покажемо алгоритм роботи програми. Закреслені клітинки зображають

відвантажені обігрівачі. У циклі розглядається найвищий стовпчик (чи кілька стовпчиків) і "зрізується" його вершина (відвантажуються обігрівачі). Загальна відвантажена кількість обігрівачів постійно порівнюється з числом *K* і в разі потреби коректується.




```
type massiv = array[0..1000000] of longint;
 Var  i,j:longint; x,y,z,n,k,kk,m:int64;
        a:massiv; f:text;
 {Процедура швидкого сортування}
 Procedure QuickSort(var a: massiv; l, r: longint);
  var m, x, y: longint;
 Begin
  m := ((l + r) div 2);
  i := l; j := r; x := a[m];
  While i <= j do
  Begin
    While a[i] < x do inc(i);
    While a[j] > x do dec(j);
    If i <= j Then
    Begin
      y:=a[i];a[i]:=a[j];a[j]:=y;
      inc(i);
      dec(j);
```

```
   End;
  End;
  If l < j Then QuickSort(a, l, j);
  If r > i Then QuickSort(a, i, r);
 End;
Begin
 Assign(f,'transit.in');Reset(f);Read(f,n,k);
 For i:=1 to n do Read(f,a[i]); Close(f);
 QuickSort(a, 1, n); {сортування масиву}
 m:=a[n]; kk:=0; y:=n; j:=1;
 While kk<k do {ідея роботи даного циклу описана вище}
 Begin
   m:=a[y-1];
   x:=a[y]-a[y-1];
   kk:=kk+x*j;
   z:=kk;
   While (z-j>=k) and (x>0) Do Begin z:=z-j;inc(m);End;
   dec(y);inc(j);
 End;
 Assign(f,'transit.out');Rewrite(f);WriteLN(f,m);Close(f);
End.
```

E - Відео-кафе Ужляндії

Input file name:	video_cafe.in
Output file name:	video_cafe.out
Time limit:	1500 ms
Memory limit:	128 M

Як відомо, чемпіонат світу з хокею у 2014 році пройде в Ужляндії. Для успішного проведення заходу приймаючій стороні належить виконати ряд вимог, що пред'являються Міжнародної хокейної федерацією. Хокейні матчі планується провести в м. Ужкачево та м. Ужковель (в Ужляндіїї всі міста починаються на Уж). Дані міста пов'язані між собою прямою автомагістраллю Ужкачево - Ужковель.

Голова Міжнародної хокейної федерації зазначив, що ні на одному з ***N*** кемпінгів, розташованих на автомагістралі Ужкачево - Ужковель немає відео-кафе. Відео-кафе - це елемент придорожнього сервісу, практично нічим не відрізняється від звичайного придорожнього кафе, але в якому створені умови для відео перегляду спортивних подій. Міжнародна хокейна федерація ухвалила, що на ***K*** з ***N*** існуючих кемпінгів необхідно побудувати відео-кафе. Міжнародна хокейна федерація також встановила такі вимоги до розташування кожного відео-кафе:

1. Якщо по дорозі з м. Ужкачево в м. Ужковель зупинитися в деякому відео-кафе, то відстань між даними відео-кафе і ***попереднім*** на автомагістралі відео-кафе (якщо таке є) повинно бути не менше ***A*** км і не більше ***B*** км.

2. Якщо по дорозі з м. Ужкачево в м. Ужковель зупинитися в деякому відео-кафе, то відстань між даними відео-кафе і ***наступним*** на автомагістралі відео-кафе (якщо таке є) повинно бути не менше ***A*** км і не більше ***B*** км.

3. Відстань від м. Ужкачево та м. Ужковель до найближчого відео-кафе не повинно бути менше ***A*** км і не більше ***B*** км.

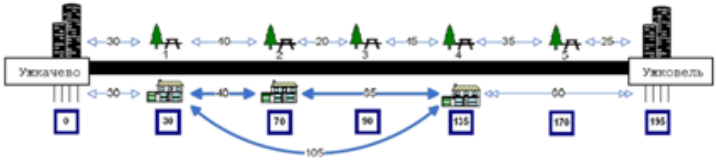


Рисунок до прикладу №1: N=5, K=3, A=20, B=70.

Для кожного ***i***-го побудованого відео-кафе введемо коефіцієнт Z_i - відстань від даного відео-кафе до всіх інших. Міжнародна хокейна федерація встановила, що чим більша сума коефіцієнтів Z_i, тим своєчасні мандрівники зможуть отримувати інформацію про проведення хокейних матчів.

Ваше завдання визначити максимальну суму Z_i, яка може бути досягнута шляхом зведення ***K*** відео-кафе, що задовольняють всім вимогам Міжнародної хокейної федерації. Відомо що, кожен кемпінг може містити на своїй території не

більше одного відео-кафе. Вірне і зворотнє - відео-кафе може розташовуватися тільки на території кемпінгу.

Формат вхідних даних: Перший рядок вхідного файлу містить два цілих числа - $N, K\ (1 \le K \le N \le 1000)$ відповідно.

Другий рядок вхідного файлу містить два цілих числа $A, B\ (1 \le A < B \le 10^9)$. Третій рядок вхідного файлу містить одне ціле число $S\ (1 \le S \le 10^9)$ - відстань між м. Ужкачево та м.Ужковель. Четвертий рядок вхідного файлу містить N цілих чисел $C_i\ (0 < C_i < S,\ C_i < C_j$ *якщо* $i < j)$ - відстань i-го кемпінгу від м.Ужкачево.

Формат вихідних даних: Вихідний файл повинен містити одне ціле число - максимальну суму коефіцієнтів Z_i побудови K відео-кафе. Рішення завжди існує.

Пояснення до прикладів:

У першому прикладі кемпінги {1, 2, 4}

У другому прикладі кемпінги {2, 5, 8, 10}.

Оцінювання:

$K \le 2$ – не менше 15 балів

$N^K \le 10^6$ – не менше 30 балів

$N \le 50$ – не менше 45 балів

$N \le 300$ – не менше 60 балів

Приклад вхідних та вихідних даних:

video_cafe.in	video_cafe.out
5 3 20 70 195 30 70 90 135 170	420
10 4 10 28 100 10 19 23 32 47 51 62 74 82 89	474

Ідея розв'язку задачі Зуба В.В., директора ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя математики, учителя-методиста; Бондаренка С.М., учителя математики та інформатики ЗОШ І-ІІІ ст. № 7 м. Прилук, учителя-методиста

Задача на реалізацію алгоритму пошуку в глибину в графі.

Розв'язок:

```
type massiv = array[1..1002] of longint;
Var  i,j,n,k,a,b,s,x,y,z,cnt:longint;zi:int64;
     c:massiv; f:text;
     vc:array[0..1001,0..1001] of boolean;
     vv:array[1..1000] of boolean;
     vq:array[1..1000] of longint;

Procedure DFS(v,x:longint; var cnt:longint); {алгоритм пошуку в глибину}
 var i,j,d,f,m,max:longint;
Begin
 m:=0;j:=0;
 For d:=1 to n do
   If vv[d]=true Then Begin j:=j+1;vq[j]:=d;inc(m);End;
 If (v=x) or (m>k) or (k=1) Then
 Begin
  If m=k Then
  Begin
    max:=0;
    For d:=1 to j-1 do
      For f:=d+1 to j do max:=max+abs(c[vq[d]]-c[vq[f]]);
    max:=max*2;      If max>=cnt Then cnt:=max;
  End;
  Exit;
 End;
```

```
  vv[v]:=true;
  For i:=1 to n+1 do
    If vc[v,i] and not vv[i] Then Begin DFS(i,x,cnt);End;
  vv[v]:=false;
 End;

Begin
 Assign(f,'videocafe.in');Reset(f);
 Read(f,n,k); Read(f,a,b); Read(f,s);
 For i:=1 to n do Read(f,c[i]);Close(f);
 c[n+1]:=s;c[n+2]:=1000000001; cnt:=0;
 {побудова матриці суміжності}
 z:=0;
 For i:=1 to n do
 Begin
  x:=z+a; y:=z+b;  j:=i;
  while (c[j]<=y) and (j<=n+1) do
  Begin
   If c[j]>=x Then vc[i-1,j]:=true;
   j:=j+1;
  End;
  z:=c[i];
 End;
 vc[i,n+1]:=true;
For i:=1 to n do If vc[0,i] Then DFS(i,n+1,cnt);
 Assign(f,'videocafe.out');Rewrite(f);WriteLN(f,cnt);Close(f);
End.
```

Програма неправильно виконує 6-12, 14-20 тести (обмеження за часом). Результат - 30 із 100.
