Управління освіти і науки
Чернігівської обласної державної адміністрації

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

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

2014-2015 НАВЧАЛЬНОГО РОКУ

Чернігів – 2016

Автори:
Бондаренко С. М., Бондарчук Н. І., Гах С. О., Дасюк А. М., Зуб В. В., Зеленський О. С., Коваленко О. І., Коробко О. М., Хорошок С. В., Хрол Н. П., Чорний А. В.

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

Загальна редакція:
Баранова О.Є., методист відділу природничо-математичних дисциплін Чернігівського обласного інституту післядипломної педагогічної освіти імені К. Д. Ушинського
Літош Ю.М., старший викладач кафедри інформатики та інформаційно-комунікаційних технологій в освіті Чернігівського обласного інституту післядипломної педагогічної освіти імені К. Д. Ушинського

Збірник задач та розв'язань ІІІ етапу Всеукраїнської учнівської олімпіади з інформатики 2014-2015 навчального року/ С. М. Бондаренко, Н. І. Бондарчук, С. О. Гах та ін.; за заг. ред. О. Є. Баранової, Ю. М. Літоша. – Чернігів: ЧОІППО імені К. Д. Ушинського, 2016. – 71 с.

Рекомендовано до друку
вченою радою Чернігівського обласного інституту
післядипломної педагогічної освіти імені К.Д. Ушинського
(протокол № 4 від 27.10.2016р.

ЗМІСТ

I тур

II тур

8-11 клас

I тур

A - Випробування автомата

Ім'я вхідного файлу: testing.in

Ім'я вихідного файлу: testing.out

Обмеження часу: 100 мс

Обмеження пам'яті: 128 M

Фірма bookface, яка створена в Ужляндії, в якій працює Степан, вирішила встановити в своїх офісах автомати з продажу чаю та кави, щоб програмісти під час перерви могли приємно провести час.

Вартість склянки чаю та кави в автоматі передбачається встановити рівній п'яти ужикам (така в Ужляндії валюта). Автомати будуть приймати монети по 5 і 10 ужиків, а також купюри в 10, 50 і 100 ужиків. Коли програмісту потрібно видавати здачу (тобто коли програміст кинув у автомат монету в 10 ужиків, або купюру в 10, 50 або 100 ужиків), автомат видає здачу монетами в п'ять ужиків; якщо ж пасажир кинув у автомат монету в п'ять ужиків, то автомат її зберігає і може використовувати для здачі наступним програмістам.

Очевидно, щоб забезпечити можливість видачі здачі всім програмістам, може знадобитися спочатку завантажити в автомат деяку кількість монет в п'ять ужиків. Зараз в офісах фірми проходять випробування з метою визначити мінімальну кількість монет, які треба завантажити в автомат перед робочим днем.

Вам дано протокол одного з таких випробувань: відомий порядок, в якому програмісти оплачували свої покупки різними монетами і купюрами. Визначте, яку мінімальну кількість монет в п'ять ужиків, повинно було спочатку перебувати в автоматі, щоб усім пасажирам вистачило здачі.

Вхідні дані: У першому рядку вхідного файлу знаходиться одне натуральне число ***N*** - кількість покупок в

автоматі, які були здійснені в ході випробування ***(1 ≤ N ≤ 50 000)***. У другому рядку знаходяться ***N*** натуральних чисел, кожне з яких рівне номіналу монети або купюри, яку використовував черговий програміст для оплати; кожен номінал може приймати одне з чотирьох значень: 5, 10, 50 або 100.

Вихідні дані: У вихідний файл виведіть одне число - мінімальну кількість монет в п'ять ужиків, які треба було завантажити в автомат спочатку, щоб усім програмістам вистачило здачі.

Примітка:

У першому прикладі одна монета в п'ять ужиків буде потрібна для здачі першому програмісту і 19 монет - третьому, але під час здачі третьому можна використовувати ту монету, яку кине другий програміст, тому спочатку у автоматі досить 19 монет.

У другому прикладі здачу третьому програмісту можна видати, використовуючи монету першого або другого покупця, і тому не потрібно завантажувати монети в автомат спочатку.

У третьому прикладі першому програмісту потрібні дев'ять монет здачі, та всі вони повинні спочатку знаходиться в автоматі.

Приклади

Вхідні дані	**Результат роботи**
3 10 5 100	19
3 5 5 10	0
4 50 5 5 5	9

Ідея розв'язання задачі

Хорошка С. В., учителя вищої категорії Коропської загальноосвітньої школи І-ІІІ ст. імені Т. Г. Шевченка Коропської районної ради

Для розв'язання задачі зручно зберігати кількість монет, які знаходяться в автоматі (ka), та кількість всіх монет (kv). Коли на зчитування приходить наступна купюра - дивимось яка вона. Якщо монета в 5 ужиків, то збільшуємо кількість монет в автоматі на 1. Для всіх інших купюр визначаємо кількість монет потрібних для здачі (k1). Якщо монет в автоматі не вистачає, то збільшуємо кількість всіх монет (kv:=kv+k1-ka), а кількість монет в автоматі обнуляємо (ka:=0), в іншому випадку зменшуємо кількість монет в автоматі (ka:=ka-k1).

Розв'язання:

```
var n,k1,j,kv,ka,i: longint;
b:int64;
begin
 assign(input,'testing.in');
reset (input);
assign(output,'testing.out');
rewrite( output);
readln(n);
kv:=0;  ka:=0;
for i:=1 to n do
begin
read(b);
if b=5 then    begin ka:=ka+1; end
      else
      begin
       b:=b-5;
       k1:=b div 5;
       if ka<k1 then  begin kv:=kv+k1-ka; ka:=0; end
              else  ka:=ka-k1;
       end;
```

```
end;
writeln( kv) ;
end.
```

Ідея розв’язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв’язання програм написані в середовищі Free Pascal 2.6.4.

Нехай на початку в автоматі монет номіналом 5 ужиків немає. При кожній покупці прораховуємо необхідну кількість монет для здачі, запам'ятовуємо результуючу кількість монет, що залишилися. Запам'ятовуємо мінімальну кількість монет. Якщо вона від'ємна, то виводимо її по модулю, інакше виводимо 0.

Розв’язання:

```
 var i,n,a:longint;k,p:int64;
Begin
 assign(input,'testing.in');reset(input);
 assign(output,'testing.out');rewrite(output);
 read(n);{кількість покупок в автоматі}
 k:=0;{кількість монеток номіналом 5 ужиків в автоматі}
 p:=0;{найменша недодатна кількість монет в автоматі}
 for i:=1 to n do
  begin
   read(a);{номінал чергової купюри/монети}
   case a of
    5:k:=k+1; {збільшення кількості монет в автоматі на 1}
    10:k:=k-1; {зменшення кількості монет в автоматі на 1}
    50:k:=k-9; {зменшення кількості монет в автоматі на 9}
```

```
    100:k:=k-19; {зменшення кількості монет в автоматі на 19}
   end;
   if (k<p) then p:=k; {порівняння поточної кількості монет в автоматі з мінімальною}
  end;
 write(abs(p)); {виведення результату}
 close(input);close(output);
End.
```

B – Кросворд

Ім'я вхідного файлу: crossword.in

Ім'я вихідного файлу: crossword.out

Обмеження часу: 1200 мс

Обмеження пам'яті: 128 M

Вам дано квадратний кросворд розміру ***N*x*N***. Порожні клітини позначені в ньому символом '-', зафарбовані - символом '#'. За правилами кросвордів, кожне слово має складатися мінімум з 2 букв. Вам потрібно для даного кросворду порахувати кількість слів по горизонталі і по вертикалі.

Вхідні дані: У першому рядку міститься число ***N (1 ≤ N ≤ 2000)*** - розмір кросворду. Наступні **N** рядків містять кросворд. Кожен рядок складається з **N** символів '-' і '#', описаних вище.

Вихідні дані: Виведіть два числа - кількість слів по горизонталі і по вертикалі.

Приклади

Вхідні дані	Результат роботи
5 ----# --##- ----- -##-- #----	5 4

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Під час підрахунку кількості слів визначаємо кількість клітинок від початку рядка до першого символу '#', між сусідніми символами '#', відстань від останнього символу до кінця рядка. У кожному випадку, коли така відстань не менша за 2, маємо чергове слово. Аналогічним чином виконуємо прорахунок для стовпців.

Розв'язання:

```
 var n,i,j,k1,k2,p:longint;
     a:array[1..2000,1..2000] of string[1];
begin
 assign(input,'crossword.in');reset(input);
 assign(output,'crossword.out');rewrite(output);
 readln(n);
 for i:=1 to n do {читання таблиці/кросворда}
  begin
    for j:=1 to n do read(a[i,j]);
    readln;
  end;
 k1:=0;k2:=0;
 for i:=1 to n do {підрахунок кількості слів по горизонталі}
  begin
    j:=0;p:=0;
    while j<n do
      begin
        inc(j);
        if a[i,j]='#'
```

```
          then begin if j-p>2 then inc(k1);p:=j;end;
        end;
      if j-p>1 then inc(k1);
    end;
   for j:=1 to n do {підрахунок кількості слів по вертикалі}
    begin
      i:=0;p:=0;
      while i<n do
        begin
          inc(i);
          if a[i,j]='#'
            then begin if i-p>2 then inc(k2);p:=i;end;
        end;
      if i-p>1 then inc(k2);
    end;
  write(k1,' ',k2);
  close(input);close(output);
End.
```

C – Цікава задача

Ім'я вхідного файлу:	interesting.in
Ім'я вихідного файлу:	interesting.out
Обмеження часу:	200 мс
Обмеження пам'яті:	128 M

Арифметична прогресія - це послідовність чисел виду ***a_1, a_1 + d, a_1*** + 2 d , a 1 ***+ (n-1)d,...***
де ***a_1*** — це перший член прогресії, ***d*** — це фіксована різниця між попереднім та наступним.

Степан, дізнавшись про проведення III етапу Всеукраїнської олімпіади з інформатики, вирішив запропонувати учасникам непросту, цікаву задачу, на тему "Арифметична прогресія". Він бере довільне додатне число ***A*** і виписує на дошці арифметичну прогресію з першим членом рівним ***A*** і різницею, рівною також A, тобто маємо послідовність

A, A+A, A+2A, A+3A, ... Степана цікавить перше число в даній послідовності, яке є повним кубом деякого натурального числа. Степан довів, що для довільного натурального числа ***A*** в описаній вище арифметичній прогресії існує повний куб деякого натурального числа.

Наприклад, перший член арифметичної прогресії 2, тоді маємо виписати на дошці послідовність 2, 4, 6, 8, ... Четвертий член цієї арифметичної прогресії є повним кубом числа 2 ($8 = 2^3$).

Напишіть програму, яка для заданого числа ***A***, визначає мінімальну кількість членів арифметичної прогресії, які потрібно виписати на дошці, щоб серед них був повний куб деякого натурального числа.

Формат вхідних даних: Єдиний рядок вхідного файлу містить одне ціле число $A(1 \le A \le 10^9)$.

Формат вихідних даних: Вихідний файл має містити одне ціле число - мінімальну кількість членів арифметичної прогресії, які потрібно виписати на дошці, щоб серед них був повний куб.

Система оцінювання:

Рішення, які вірно працюють для $A \le 10$ набиратимуть не менше 20 балів.

Пояснення: Перший приклад: четвертий член цієї арифметичної прогресії (2, 4, 6, 8, ...) є повним кубом числа 2 ($8 = 2^3$).

Другий приклад: другий член цієї арифметичної прогресії (4, 8, ...) є повним кубом числа 2 ($8 = 2^3$).

Третій приклад: перший член цієї арифметичної прогресії (125, ...) є повним кубом числа 5 ($125 = 5^3$).

Приклади

Вхідні дані	Результат роботи
2	4
4	2
125	1

Ідея розв'язання задачі

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

Розкладемо число N на прості множники. Отримаємо: $N = p_1^{s1} * p_2^{s2} * p_3^{s3}...p_k^{sk}$. Де p_i - просте число.

Очевидно, що найближчий куб до числа N, це таке число M, яке ми отримали як N помножене на відповідне число X. Щоб отримати M, нам потрібно, щоб для всіх множників(простих чисел) показники степеня були кратні трьом.

Тому будемо робити таким чином:

X = 1;

Для кожного числа p_i, якщо s_i при діленні на 3 дає остачу:

0 - X змінювати не треба.

1 - то потрібно X домножити на p_i^2, щоб при множенні X * N p_i було в степені кратній трьом.

2 - то потрібно X домножити на p_i, з тих самих міркувань.

Розв'язання:

```
var a,i,x,s,n:int64;
    flag:boolean;
begin
assign(input,'interesting.in');
assign(output,'interesting.out');
reset(input);
rewrite(output);
read(a);
n:=a;
i:=2;
x:=1;
flag:=false;
while i<=n div 2 do
  begin
  if  a mod i=0 then begin
```

```
                s:=0;
                flag:=true;
                while a mod i=0 do
                  begin
                  s:=s+1;
                  a:=a div i;
                  end;
                if s mod 3=1  then x:=x*i*i;
                if s mod 3 =2  then x:=x*i;
                i:=i+1;
                if a=1 then break;
                end
             else i:=i+1;
 if (i>10000000) and (flag=false)  then break;
 if (i>10000000) and (a>10000000) then begin x:=x*a*a; break; end;
  end;
if flag=false then write(a*a) else write(x);
close(input);
close(output);
end.
```

Ідея розв'язання задачі

Хорошка С. В., учителя вищої категорії Коропської загальноосвітньої школи І-ІІІ ст. імені Т. Г. Шевченка Коропської районної ради

Потрібно розкласти число на прості множники. Найближчий куб до числа **N**, це таке число **M**, яке ми отримали як **N** помножене на відповідне число **X**. Щоб отримати М, нам потрібно, щоб для всіх множників (простих чисел) показники степеня були кратні трьом.

Для розв'язку задачі будемо зберігати кількість простих множників у масиві А.

Якщо а[і] відмінне від нуля, то є три варіанти:

остача від ділення на 3=0 – то шукане число М домножаємо на і (просте число) a[i] раз.
остача від ділення на 3=1– то шукане число М домножаємо на і (просте число) a[i] +2 раз.
остача від ділення на 3=2 – то шукане число М домножаємо на і (просте число) a[i] +1 аз.

Числом М, поділене на наше початкове число, і дасть мінімальну кількість членів арифметичної прогресії.

Розв'язання:

```
program c;
var
n, p: longint;   k,n1,i,j:longint;
a:array [1..10000000] of integer;
m,d,b3,max:int64;
begin
assign(INPUT,'interesting.in');
reset (input);
assign(output,'interesting.out');
rewrite( output);
fillchar(a,sizeof(a),0);
readln(n);  n1:=n;
p := 2; max:=p;
while n <> 1 do begin
if (n mod p) = 0 then begin a[p]:=a[p]+1; max:=p;
n := n div p
end
else inc(p)
end ;
m:=1 ;
for i:=1 to max do
begin
if a[i]<>0 then if a[i] mod 3 =0 then begin
                                  for j:=1 to a[i] do
                                    m:=m*i
                                    end
```

```
else   if a[i] mod 3 =1 then begin
                                for j:=1 to a[i]+2 do
                                  m:=m*i
                                  end
                          else begin
                                for j:=1 to a[i]+1 do
                                  m:=m*i
                                  end;
end;
writeln(m div n1);
end.
```

На сервірі програма правильно виконує 18 тестів на 90 балів (2 тести ліміт часу).

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Розв'язання на 55 балів

Перевірятимо члени прогресії за формулою $\left(\sqrt[3]{A}\right)^3 = A$.

Розв'язання:

```
 var a,b,c:real;i:int64;
begin
  assign(input,'interesting.in');
 reset(input);
 assign(output,'interesting.out');
 rewrite(output);
 read(a);
```

```
  i:=0;b:=0;c:=a;
  while b*b*b<>c do
    begin
      inc(i);
      c:=a*i;
      b:=int(exp(ln(c)/3));
    end;
  write(i);
  close(input);
  close(output);
end.
```

Розв'язання на 80 балів

Розкладемо число ***A*** на прості множники: $A=2^{n1}3^{n2}5^{n3}\ldots$. Якщо числа ***n1, n2, n3,*** … кратні 3, то їх можна відкинути (замінити на 0). В іншому випадку спочатку знайти їх остачу від ділення на 3, потім відняти від 3. Знайдене число записати замість початкового. Отримане таким чином число і буде шуканим. Наприклад, $2=2^1$. Число ***1*** не кратне ***3***, остача від ділення на ***3*** буде ***1***, ***3-1=2***. Нове число буде $2^2=4$.

Розв’язання:

```
var a,i,k,s:int64;
begin
  assign(input,'interesting.in');reset(input);
  assign(output,'interesting.out');rewrite(output);
  read(a);
  i:=2; s:=1;
  while a>1 do
    begin
      k:=0;
      while a mod i =0 do begin a:=a div i;k:=k+1;end;
      if k<>0 then
        begin
          k:=k mod 3;
          if k=1 then s:=s*i*i;
```

```
        if k=2 then s:=s*i;
       end;
     i:=i+1;
   end ;
   write(s);
   close(input);close(output);
end.
```

Розв'язання на 100 балів.

Ідея розв'язання полягає в знаходженні першого куба числа, яке ділиться на число A.
(i:=1; while i*i*i mod a<>0 then inc(i); write(i*i*i div a);

Розв’язання:

```
 var i,n,s,k,x,y:int64;
Begin
  Assign(input,'interesting.in');Reset(input);
  Read(n);Close(input);
  i:=2; k:=2; x:=0; s:=1; y:=1;
  While n <> 1 do
    Begin
      While (n mod i <> 0) and (i<=sqrt(n)) do inc(i,y); {Визначення
першого дільника числа n, крім самого числа n}
      If n mod i <> 0 Then i:=n; {Якщо дільник відсутній, то ним є
саме число n}
      If i<>k Then
        Begin
          If x mod 3=1 Then s:=s*k*k;
          If x mod 3=2 Then s:=s*k;
          x:=1;
          k:=i;
          y:=2;
        End
       Else x:=x+1;
      n:= n div i;
    End;
```

```
  If x mod 3=1 Then s:=s*i*i;
  If x mod 3=2 Then s:=s*i;
  Assign(output,'interesting.out');Rewrite(output);
  WriteLN(s);Close(output);
End.
```

D – Увімкніть лампу

Ім'я вхідного файлу: lamp.in

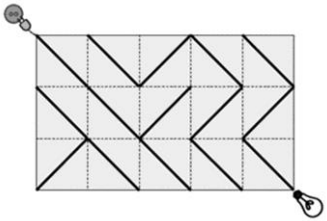
Ім'я вихідного файлу: lamp.out

Обмеження часу: 1000 ms

Обмеження пам'яті: 128 M

Степан розробляє електронну схему на прямокутній сітці розміром ***N*M***. Усього ***N*M*** квадратних плиток. Два (з чотирьох) протилежних кута кожної плитки з'єднані дротом.

Джерело живлення під'єднано до лівого верхнього кута сітки, лампа – до правого нижнього. Для того, щоб увімкнути лампу, можна повернути будь-яку плитку на 90 градусів у обох напрямках.



На зображені лампа вимкнута. Якщо повернути будь-яку плитку у другій справа колонці, то лампа увімкнеться.

Напишіть програму, яка знаходить мінімальну кількість плиток, що треба перевернути для того, щоб увімкнути лампу.

Вхідні дан: Перший рядок містить два цілих числа ***N*** та ***M*** – розміри сітки. Далі слідують ***N*** рядків по ***M*** символів – \ або /, що характеризують напрямлення дроту на даній плитці.

Вихідні дані: Єдиний рядок вихідного файлу має містити відповідь на задачу, або ж повідомлення "NO SOLUTION", в тому випадку, коли увімкнути лампу неможливо.

Обмеження: 1 ≤ N, M ≤ 500.

Система оцінювання: В даній задачі дванадцять блоків. Тести нараховуються окремо за кожен блок, при умові проходження усіх тестів блоку.
40 балів можна отримати, якщо Ваша програма вірно працює при $1 \le N \le 4$ та $1 \le M \le 5$.

Приклади

Вхідні дані	Результат роботи
3 5 \/\\ \// /\\\	1

Ідея розв'язання задачі

Зеленського О. С., учителя інформатики та фізики, учителя-методиста Куликівської загальноосвітньої школи І-ІІІ ст. Куликівської районної ради

Побудуємо по цій прямокутній сітці граф, де кути плиток будуть вершинами. Для кожної плитки ребра будуватимуться таким чином: якщо з лівого верхнього кута можна потрапити в правий нижній, не повертаючи плитку, то між відповідними вершинами будуємо ребро вартості 0, якщо плитку потрібно повертати, то будуємо ребро вартості 1.

Аналогічно для випадку з правого верхнього в лівий нижній куток.

Потім на цьому графі знаходимо найкоротший шлях (наприклад алгоритмом Дейкстри) з вершини, яка відповідає за джерело живлення, в ту, яка відповідає за лампочку.

Вартість цього шляху і буде відповіддю на задачу.

Так як граф отримуємо розріджений (кількість ребер значно менша квадрату кількості вершин), то представимо граф у вигляді зв'язаного списку, розміщеного в динамічній пам'яті.

Розв'язання:

```
const
  maxn = 501*501;
  inf  = 223372036854775807; //максимально можливе для int64. Нескінченність

type
  pnode = ^tnode;
  tnode = record
    v: int64;
    weight: 0..1;
    next: pnode;
  end;


var
  a: array [1..maxn] of pnode;
  d: array [1..maxn] of int64;
  //p: array [1..maxn] of int64;
  visited: array [1..maxn] of boolean;
  s: int64;
  n, M: int64;
  start,fin,x:int64;
  i,j: int64;
  lv,ln,pv,pn:int64;

//Процедура створення вершини в списку
procedure insert_vertex(v, w: int64; weight: int64);
var
  p: pnode;
begin
  new(p);
  p^.v := w;
  p^.weight := weight;
  p^.next := a[v];
  a[v] := p;
```

```
end;


procedure init;
var
  i, j, x, y, nn, z: int64;
  c:char;
begin

 for i := 1 to maxn do
  begin
    a[i] := nil;
    d[i] := inf;
  end;

  //fillchar(p, sizeof(p), 0);
  fillchar(visited, sizeof(visited), false);
          //Читаємо висоту та ширину прямокутної сітки
          readln(N,M);
//Розрахуємо кількість вершин. fin — номер останньої вершини.
//Усі вершини розміщені на перетині кутів плиток.
          fin:=(n+1)*(m+1);
          for i:=1 to N do begin
                    for j:=1 to M do begin
                              //Номер лівої верхньої вершини
                              lv:=i*(M+1)-(M+1)+j;
                              //Номер правої нижньої вершини
                              pn:=(i+1)*(M+1)-(M+1)+(j+1);
                              //Номер лівої нижньої вершини
                              ln:=(i+1)*(M+1)-(M+1)+j;
                              //Номер правої верхньої вершини
                              pv:=i*(M+1)-(M+1)+(j+1);
                              Read(c);
// при цьому символі між чотирма вершинами
//у ребра “\” встановимо вагу 0,  а в ребра “/”- вагу 1.

                              if c='/' then begin
```

```
                                insert_vertex(lv, pn, 1);
                                insert_vertex(pn, lv, 1);
                                insert_vertex(pv, ln, 0);
                                insert_vertex(ln, pv, 0);
                        end
//у протилежному випадку для “\” встановимо 1,
//а для  “/”- вагу 0.
                        else begin
                                insert_vertex(lv, pn, 0);
                                insert_vertex(pn, lv, 0);
                                insert_vertex(pv, ln, 1);
                                insert_vertex(ln, pv, 1);
                        end;
                end;
                Readln;
        end;
end;

//Алгоритм Дейкстри для графу, заданого зв'язаним списком
//Деякі команди закоментовано, так як маршрут за умовами задачі знаходити не потрібно
procedure dijkstra(s: int64);
var
  j, i, min, v, weight: int64;
  t: pnode;
begin
  t := a[s];
  while t <> nil do
  begin
    d[t^.v] := t^.weight;
    t := t^.next;
  end;

  //for v := 1 to fin do
 //   p[v] := s;
  d[s] := 0;
  //p[s] := 0;
```

```
  visited[s] := true;

  for j := 2 to fin do begin
    min := inf;
    for i := 1 to fin do
      if (not visited[i]) and (d[i] < min) then begin
        min := d[i];
        v := i;
      end;
    t := a[v];
    visited[v] := true;
    while t <> nil do begin
      if d[t^.v] > d[v] + t^.weight then begin
        d[t^.v] := d[v] + t^.weight;
       // p[t^.v] := v;
      end;
      t := t^.next;
    end;
  end;
end;


begin
        assign(input,'lamp.in');
        reset(input);
        assign(output,'lamp.out');
        rewrite(output);
        init;
//Викликаємо алгоритм Дейкстри для 1 вершини
//(з неї шукатимемо найкоротший шлях до всіх інших вершин.
        dijkstra(1);
//Виводимо результати роботи програми
        if d[fin]<>inf then
                writeln(d[fin])
        else
                writeln('NO SOLUTION');
```

```
        //Очищуємо пам'ять від масиву в динамічній пам'яті
        for i := 1 to fin do dispose(a[i]);
        close(input);
        close(output);

end.
```

Програма виконує 34 тести із 53 тестів.

Ідея розв'язання задачі

Дасюка Антона, учня 11 класу спеціалізованої загальноосвітньої школи № 2 I-III ст. з поглибленим вивченням іноземних мов м. Чернігова;

Коваленко О.І., учителя-методиста, учителя інформатики спеціалізованої загальноосвітньої школи № 2 I-III ст. з поглибленим вивченням іноземних мов м. Чернігова

Розв'язання:

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<queue>
using namespace std;
#define endl "\n"
const int INF = 1000000000;
int main()
{
   int n,m,k=0;
   cin>>n>>m;
   string str;
   char a[n+5][m+5];
   int num[n+5][m+5];
   /**
      Зчитування вхідних даних
   **/
```

```
for (int i=1;i<=n;++i)
{
   cin>>str;
   for (int j=1;j<=m;++j)
      a[i][j]=str[j-1];
}
/**
   Представимо граф, в якому вершини, це точки плиток, а
ребра, перехід по діагоналям.
   Пронумеруємо всі вершини
**/
for (int i=1;i<=n+1;++i)
   for (int j=1;j<=m+1;++j)
      num[i][j]=++k;
int Size = (n+1)*(m+1)+2;
/**
   Граф будемо зберігати у вигляді списку ребер формату
(куди веде, відстань).
   Якщо з однієї вершини в іншу можна дійти не
перемикаючись, будемо вважати, що відстань між ними = 0,
інакше 1.
**/
vector < vector <pair<int,int> > > g(Size);
for (int i=1;i<=n;++i)
   for (int j=1;j<=m;++j)
   {
      int tmp = 0;
      if(a[i][j]=='\\')
         tmp = 1;
      g[num[i][j]].push_back(make_pair(num[i+1][j+1],1-tmp));
      g[num[i+1][j+1]].push_back(make_pair(num[i][j],1-tmp));
      g[num[i+1][j]].push_back(make_pair(num[i][j+1],tmp));
      g[num[i][j+1]].push_back(make_pair(num[i+1][j],tmp));
   }
/**
```

За допомогою алгоритму Дейкстри, знайдемо мінімальну відстань від вершини (1,1) до всіх інших.

У випадку якщо дістатися до вершини неможливо, будемо зберігати достаньо велике число, яке не зможемо набрати за умовою.

```
   **/
   int s = 1;
   vector<int> d (Size, INF),  p (Size);
   d[s] = 0;
   priority_queue < pair<int,int> > q;
   q.push (make_pair (0, s));
   while (!q.empty()) {
      int v = q.top().second,  cur_d = -q.top().first;
      q.pop();
      if (cur_d > d[v])  continue;
      for (size_t j=0; j<g[v].size(); ++j) {
         int to = g[v][j].first,
            len = g[v][j].second;
         if (d[v] + len < d[to]) {
            d[to] = d[v] + len;
            p[to] = v;
            q.push (make_pair (-d[to], to));
         }
      }
   }
   /**
      Виводимо відповідь
   **/
   if(d[num[n][m]]==INF)
      cout<<"NO SOLUTION"<<endl;
   else
      cout<<d[num[n+1][m+1]]<<endl;
   return 0;
}
```

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Пошук мінімального шляху в графі за допомогою алгоритму Дейкстри.

Розв'язання:

```
var i,j,n,m,s,r:longint; st:string[1];
    a:array[0..502,0..502] of longint; Ok,Ok1:boolean;
    v:array[0..502,0..502] of boolean;
    c:array[0..502,0..502] of longint;
    t:array of array of longint;

Function PN(x:longint):boolean;
  Begin
    If x mod 2=0 Then PN:=True Else PN:=false;;
  End;
{Визначення мінімальної кількості повернутих плиток для увімкнення лампи}
Function Deykstra(aa,bb:longint):longint;
  Var x,y,mm,rr,xt,yt:longint;
Begin
  If n>=m Then r:=n Else r:=m;
  SetLength(t,r+1); For i:=0 to r do SetLength(t[i],1);
  For i:=1 to n+1 do For j:=1 to m+1 do c[i,j]:=1000000000;
  v[1,1]:=true;
  c[aa,bb]:=a[1,1];
  mm:=c[aa,bb];
  inc(t[mm,0]);
  SetLength(t[mm],t[mm,0]+1);
```

```
t[mm,t[mm,0]]:=1000*2+2;
xt:=c[aa,bb]; yt:=1;

While c[n+1,m+1]=1000000000 do
 Begin
  x:=t[xt,yt] div 1000;
  y:=t[xt,yt] mod 1000; v[x,y]:=true;
  If (not v[x-1,y-1]) and (c[x-1,y-1]>c[x,y]+a[x-1,y-1])
   Then
    Begin c[x-1,y-1]:=c[x,y]+a[x-1,y-1];
      rr:=c[x-1,y-1];inc(t[rr,0]);
      SetLength(t[rr],t[rr,0]+1);
      t[rr,t[rr,0]]:=(x-1)*1000+y-1;
    End;
  If (not v[x+1,y-1]) and (c[x+1,y-1]>c[x,y]+a[x,y-1])
   Then
    Begin c[x+1,y-1]:=c[x,y]+a[x,y-1];
      rr:=c[x+1,y-1];inc(t[rr,0]);
     SetLength(t[rr],t[rr,0]+1);
     t[rr,t[rr,0]]:=(x+1)*1000+y-1;
    End;
  If (not v[x-1,y+1]) and (c[x-1,y+1]>c[x,y]+a[x-1,y])
   Then
    Begin c[x-1,y+1]:=c[x,y]+a[x-1,y];
      rr:=c[x-1,y+1]; inc(t[rr,0]);
      SetLength(t[rr],t[rr,0]+1);
      t[rr,t[rr,0]]:=(x-1)*1000+y+1;
    End;
  If (not v[x+1,y+1]) and (c[x+1,y+1]>c[x,y]+a[x,y])
   Then
    Begin c[x+1,y+1]:=c[x,y]+a[x,y];
      rr:=c[x+1,y+1];inc(t[rr,0]);
      SetLength(t[rr],t[rr,0]+1);
      t[rr,t[rr,0]]:=(x+1)*1000+y+1;
    End;
  inc(yt);If yt>t[xt,0] Then Begin yt:=1; inc(xt);End;
```

```
    End;
   Deykstra:=c[n,m]+a[n,m];
  End;

Begin
  Assign(Input,'lamp.in');Reset(Input);Readln(n,m);
  ok:=(not PN(n) and not PN(m)) or (PN(n) and PN(m));
  If Ok Then s:=0 Else s:=1;
  Assign(Output,'lamp.out');Rewrite(Output);
{Визначення плиток,  які необхідно повертати для увімкнення лампочки}
  For i:=1 to n do
   Begin
     If s=1 Then Break;
     ok1:=PN(i);
     For j:=1 to m do
      Begin
        Read(st);
        ok:=((not Ok1) and (not PN(j)) or Ok1 and PN(j) ) ;
        If  ok and (st<>'\') Then a[i,j]:=1;
        If  (not ok) and (st<>'/') Then a[i,j]:=1;
      End;
     ReadLN();
   End;
  Close(Input);
  If s=1 Then WriteLN('NO SOLUTION')
        Else WriteLN(Deykstra(2,2));
  Close(Output);
End.
```

E – Свято в Ужляндії

Ім'я вхідного файлу: holiday.in

Ім'я вихідного файлу: holiday.out

Обмеження часу: 1500 ms

Обмеження пам'яті: 128 M

В Ужмісті – столиці Ужляндії, планується провести фестиваль Ужляндських виробів. Президент Ужляндії розуміє, що гості з усіх куточків Ужляндії приїдуть на цей фестиваль і хоче зробити їх подорож як можна менш витратною.

Ужляндія складається з N міст. Дорожня система Ужляндії не змінювалася з давніх пір, і тому вона містить лише $N-1$ двонапрямлених доріг. По цих дорогах все ще можна проїхати з будь-якого міста Ужляндії в будь-яке інше (можливо, через проміжні міста). Історично за проїзд по кожній дорозі стягується податок. Він стягується одним з міст, що з'єднуються дорогою. Кажуть, що це місто відповідальне за цю дорогу. Величина податку може бути різною для різних доріг.

Для проїзду до столиці деяким жителям Ужляндії необхідно заплатити більше, а іншим менше. Це впливає на атмосферу свята: той, кому довелося заплатити занадто багато, приїжджає без настрою. Президент хоче мінімізувати суму дорожнього податку, яку потрібно заплатити на шляху до столиці з будь-якого міста Ужляндії. Для цього він вирішив наказати усім містам скасувати податок на одній з доріг, за які вони відповідальні. Якщо місто не відповідальне за жодну дорогу, то і скасовувати податок йому не потрібно.

Допоможіть визначити, на яких дорогах потрібно скасувати податок, щоб вартість подорожі до столиці стала якомога менша. Вартість подорожі до столиці визначається як максимальне значення для всіх міст Ужляндії.

Вхідні дані: Перший рядок вхідного файлу містить ціле число N $(2 \le N \le 10^5)$ – кількість міст в Ужляндії. Міста пронумеровані натуральними числами від 1 до N. Ужмісто має номер 1. Далі йдуть $N-1$ рядок. Кожен з цих рядків містить три

розділених одиночними пробілами цілих числа ***X, Y, Z:*** між містами з номерами ***X*** і ***Y*** є дорога, за яку відповідальне місто з номером X, і за проїзд по якій стягується податок рівний ***Z*** ($1 \le Z \le 10^4$).

Вихідні дані: Єдиний рядок вихідного файлу має містити одне ціле число – мінімально можливе значення суми податків на шляху з міста в столицю.

Система оцінювання:

$N \le 10$ - не менше 20 балів

$N \le 20$ - не менше 30 балів

$N \le 200$ - не менше 50 балів

$N \le 2000$ - не менше 70 балів

Приклади

Вхідні дані	**Результат роботи**
3 1 2 2 1 3 3	2
5 1 2 1 1 3 1 3 4 1 3 5 1	1

Ідея розв'язання задачі

Дасюка Антона, учня 11 класу спеціалізованої загальноосвітньої школи № 2 І-ІІІ ст. з поглибленим вивченням іноземних мов м. Чернігова;

Чорного Андрія, учня 9 класу Чернігівського колегіуму №11 Чернігівської міської ради;

Коробко О. М., учителя інформатики Чернігівського колегіуму №11 Чернігівської міської ради

Ця задача розв'язується за допомогою динамічного програмування на дереві. Розглядатимемо граф як дерево не зважаючи на напрямок руху(ніби граф неорієнтований), але будемо пам`ятати про напрямок ребер.

Розділимо вершини на дві категорії:

Чорні - в яких ребро, яке поєднує вершину і її батька, має напрямок "до батька"

Білі - в яких ребро, яке поєднує вершину і її батька, має напрямок "від батька".

Заведемо масив d[v][0..1], де v - номер вершини.

В d[v][0] буде зберігатись відповідь на задачу для піддерева з коренем у вершині v(для чорних вершин відповідь, для якої з ребра, яке поєднує цю вершину і її батька, не знімається податок.

В d[v][1] в білих вершинах буде 0, а в чорних буде зберігатись відповідь на задачу для піддерева з коренем у вершині v, але з ребра, яке поєднує цю вершину і її батька, було виключено податок.

Проходимо по дереву, починаючи з листків, тобто вершин, для яких не існує вершин нижчого рівня, для них вартість шляху буде 0.

Далі перебираємо вершини, рухаючись у напрямку до першої (столиці), для кожної знаходимо 2 максимуми вартості шляху до них з вершин нижчого рівня. Далі перший максимум ми можемо "прибрати", відмінивши податок з відповідної дороги, а «другий максимум» буде включений до відповіді.

Розв'язання:

```
#include<iostream>
#include<vector>
#include<algorithm>
#include<cstdio>
#include<cstring>
using namespace std;
#define pb push_back
#define mp make_pair
#define X first
#define Y second
#define fi(st,n) for (int i = (st); i < (n); i++)
typedef pair<int, int> pii;
const int inf = (int) 1e9, maxn = (int) 1e5 + 5;
vector < pair < pii, int > > g[maxn+5];
int d[maxn], from, to, cost, n;;
void solve(int v, int cost, int f, bool notMY)
{
   for (int to = 0; to < g[v].size(); ++ to)
      if(g[v][to].X.X!=f)
            solve(g[v][to].X.X, g[v][to].X.Y, v, g[v][to].Y);
   int max1 = 0, max2 = 0;
   for (int to = 0; to < g[v].size(); ++ to)
      if(g[v][to].Y==1&&g[v][to].X.X!=f)
      {
         int tmp = d[g[v][to].X.X]+g[v][to].X.Y;
         if(tmp>=max1)max2=max1, max1=tmp;
         else if(tmp>max2) max2=tmp;
      }
   if (notMY)
      d[v]=max(d[v],max2);
   else
      d[v]=max(d[v],min(max2+cost, max1));
   for (int to = 0; to < g[v].size(); ++ to)
      if(g[v][to].X.X!=f)
            d[v] = max(d[v],d[g[v][to].X.X]);
```

```
    return;
}
int main()
{
    freopen("holiday.in", "r", stdin);
    freopen("holiday.out", "w", stdout);
    memset(d, 0, sizeof(d));
    cin>>n;
    for (int i = 1; i < n; ++ i)
    {
        cin>>from>>to>>cost;
        g[from].pb( mp( mp( to, cost ), 1));
        g[to].pb( mp( mp( from, cost ), 0));
    }
    solve(1, 0, 0, 1);
    cout<<d[1];
          return 0;
}
```

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Розв'язання:

```
Type massiv=array[1..100000,1..3]  of longint;
     mas=array[1..100000] of longint;

var i,j,n,k,m,b,code:longint; x,y,z,jj:int64; s1,s2:string;
    a,v:massiv; c,d,p,r:mas;
```

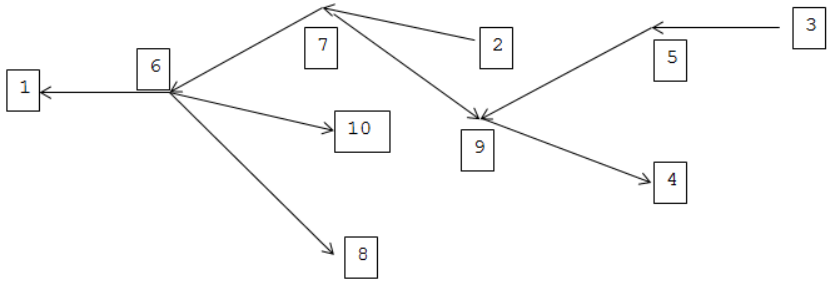
```
    st:array[1..100000] of string;

  Procedure abc;
   Begin
    If c[a[j,1]]>0 Then dec(c[a[j,1]]);
    If p[a[j,2]]>0 Then dec(p[a[j,2]]);
    If c[a[j,1]]+p[a[j,1]]=1
     Then Begin r[m]:=a[j,1];m:=m+1;End;
    If c[a[j,2]]+p[a[j,2]]=1
     Then Begin r[m]:=a[j,2];m:=m+1;End;
  End;

Begin
  Assign(Input,'holiday.in');Reset(Input);Readln(n);
  For i:=1 to n-1 do
   Begin
    ReadLN(x,y,z);
    a[i,1]:=x; a[i,2]:=y;a[i,3]:=z; {Побудова таблиці даних}
    c[x]:=c[x]+1; {Підрахунок кількості доріг,  за які відповідає
місто X}
    d[x]:=d[x]+1;
    p[y]:=p[y]+1; {Підрахунок кількості доріг, які ведуть у місто
Y}
    Str(i,s1); st[x]:=st[x]+s1+'.'; st[y]:=st[y]+s1+'.';
{Таблиця місцезнаходження даних у таблиці A для міста X або
міста Y}
  End;
```

A	5	7	7	3	2	6	6	9	6
	9	9	6	5	7	10	8	4	1
	405	2189	6883	6717	3145	3872	9359	2295	6232

Міста	1	2	3	4	5	6	7	8	9	10
C, D		1	1			3	2		1	

Міста	1	2	3	4	5	6	7	8	9	10
P	1			1	1	1	1	1	2	1

Міста	1	2	3	4	5	6	7	8	9	10
ST	9.	5.	4.	8.	1.4.	3.6.7.9.	2.3.5.	7.	1.2.8.	6.




```
 Close(Input);
{Визначення мінімально можливого значення суми податків на шляху з міст в столицю}
 Assign(Output,'holiday.out');Rewrite(Output);
 m:=1;
 For i:=1 to n do
  If c[i]+p[i]=1 Then Begin r[m]:=i; m:=m+1;End;
 For b:=1 to n do
  Begin
   i:=r[b]; If (i=1) and (b<=n) Then Continue;
   val(copy(st[i],1,length(st[i])-1),j,code);
   Str(j,s1); s1:=s1+'.';

   If i=a[j,1]
    Then Delete(st[a[j,2]],Pos(s1,st[a[j,2]]),length(s1));
   If i=a[j,2]
    Then Delete(st[a[j,1]],Pos(s1,st[a[j,1]]),length(s1));
   If ((i=a[j,1]) and (d[a[j,1]]=1)) or ((i=a[j,2])
    and (d[a[j,1]]=1)) Then a[j,3]:=0;

   If (c[a[j,2]]=0) and (p[a[j,2]]=1) and (a[j,2]<>1)
```

```
 Then
  Begin
   If v[a[j,1],2]<a[j,3]+v[a[j,2],1]
    Then
     Begin
      If v[a[j,1],2]>v[a[j,1],3]
       Then v[a[j,1],3]:=v[a[j,1],2];
      v[a[j,1],2]:=a[j,3]+v[a[j,2],1];
      If v[a[j,2],1]>v[a[j,1],3]
       Then v[a[j,1],3]:=v[a[j,2],1];
     End
    Else If v[a[j,1],3]<a[j,3]+v[a[j,2],1]
         Then v[a[j,1],3]:=a[j,3]+v[a[j,2],1];
   abc;
   If (c[a[j,1]]=0) and  (v[a[j,1],1]<v[a[j,1],3])
    Then v[a[j,1],1]:=v[a[j,1],3];

   a[j,1]:=0; a[j,2]:=0; a[j,3]:=0;
  End;

If (c[a[j,1]]=1) and (p[a[j,1]]=0) and (a[j,1]<>1)
 Then
  Begin
   If v[a[j,1],2]<>0
    Then
     Begin
      If v[a[j,1],2]<v[a[j,1],1]
       Then k:=v[a[j,1],1] Else k:=v[a[j,1],2];
      If v[a[j,1],3]<v[a[j,1],1]
       Then v[a[j,1],3]:=v[a[j,1],1];
      If a[j,3]+v[a[j,1],3]<k Then k:=a[j,3]+v[a[j,1],3];
      If k>v[a[j,2],1] Then v[a[j,2],1]:=k;
     End
    Else If v[a[j,2],1]<a[j,3]+v[a[j,1],1]
         Then v[a[j,2],1]:=a[j,3]+v[a[j,1],1];
   abc;
```

```
      a[j,1]:=0; a[j,2]:=0;a[j,3]:=0;
     End;

  End;
 WriteLN(v[1,1]);Close(Output);
End.
```

II тур

A – День іменинника

Ім'я вхідного файлу: birthday.in

Ім'я вихідного файлу: birthday.out

Обмеження часу: 200 мс

Обмеження пам'яті: 128 M

Під час проведення I Всеукраїнської Зимової Комп'ютерної школи було вирішено відсвяткувати – "День іменинника". ***K*** учасників святкували свій день народження, а інші ***N*** вирішили зробити іменинникам солодкий сюрприз і принесли кожен по шоколадці і віддали їх старшій вожатій Марині (усі знають що, для Марини головне – чесність), щоб вона поділила їх між іменинниками. Марина хоче роздати шоколадки іменинникам так, щоб у кожного з них був однаковий набір шоколадок: однакова кількість молочних шоколадок, шоколадок із ізюмом, чорного шоколаду і інших. Може вийти так, що усі шоколадки порівну розділити не вийде. У цьому випадку їх можна буде використовувати в якості призів на конкурсах.

Марина хоче роздати якомога більше шоколадок іменинникам, проте вона зайнята підготовкою конкурсів. Вона просить Вас визначити оптимальний набір шоколадок, який отримає кожен з ***K*** іменинників.

Вхідні дані: Перший рядок вхідного файлу містить два числа N, K $(1 \le N, K \le 10^5)$ – кількість шоколадок і кількість іменинників відповідно. Другий рядок вхідного файлу містить ***N*** цілих чисел a_i $(1 \le a_i \le 100)$ – кількість шоколадок кожного типу.

Вихідні дані: Перший рядок вихідного файлу має містити число ***M*** – максимальна кількість шоколадок, яку зможе отримати кожен іменинник. Другий рядок повинен містити ***M*** цілих чисел b_i – типи шоколадок (з урахуванням кількості), які отримає кожен іменинник. Дана послідовність має бути виведена в порядку не спадання.

Система оцінювання:

N ≤ 100, для усіх ***I: a_i = 1*** – не менше 20 балів

N ≤ 100, для усіх ***I: a_i ≤ 2*** – не менше 40 балів

N ≤ 5000 – не менше 75 балів.

Пояснення: Перший приклад. Маючи 4 шоколадки першого типу і 3 другого типу, обидва іменинника отримають по дві шоколадки першого типу і по одній другого. Другий приклад. Усі шоколадки будуть використані в якості призів.

Приклади

Вхідні дані	Результат роботи
7 2 1 2 1 2 1 2 1	3 1 1 2
5 2 1 2 3 4 5	0

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи I-III ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи I-III ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Визначення кількості шоколадок кожного виду, та визначення їх кількості для кожного іменинника (формування подарунка).

Розв'язання:

```
 var i,j,n,k:longint; x,y,c:int64; f:text;
     a,b:array[1..100] of longint;
Begin
```

```
  Assign(f,'birthday.in');Reset(f);Readln(f,n,k);
  y:=1;
  For i:=1 to n do {Визначення числа шоколадок кожного виду та кількості видів шоколадок}
   Begin
    Read(f,x);
    If x>y Then y:=x;
    b[x]:=b[x]+1;
   End;
  Close(f);
  c:=0;
  Assign(f,'birthday.out');Rewrite(f);
  For i:=1 to y do {Визначення максимальної кількості шоколадок кожного виду, яку може отримати іменинник}
   Begin
    a[i]:=b[i] div k;
    c:=c+a[i];
   End;
  WriteLN(f,c);
  For i:=1 to y do
   If a[i]>0 Then For j:=1 to a[i] do Write(f,i,' ');
  Close(f);
End.
```

B – Стрічка

Ім'я вхідного файлу: ribbon.in

Ім'я вихідного файлу: ribbon.out

Обмеження часу: 200 мс

Обмеження пам'яті: 128 M

У Степана є стрічка довжини ***N***. Він хоче розрізати її так, щоб виконувалися дві умови:

- після розрізання, кожна частина стрічки повинна бути довжиною ***a, b*** або ***c***;

– кількість частин стрічки після розрізання повинна бути якомога більше.

Допоможіть Степану, знайдіть кількість частин стрічки після необхідного розрізання.

Вхідні дані: У першому рядку записано через пропуск чотири цілих числа ***N a, b, c (1 ≤ N, a, b, c ≤ 4000)*** – довжина заданої стрічки і дозволені довжини частин стрічки після розрізання, відповідно. Числа ***a, b*** і ***c*** можуть збігатися.

Вихідні дані: Виведіть одне число – максимально можливу кількість частин стрічки. Гарантується, що існує хоча б одне коректне розрізання стрічки.

Пояснення: Перший приклад: потрібно розрізати стрічку на дві частини: одна з них довжиною 2, друга довжиною 3.Другий порядок: потрібно розрізати стрічку на два частини: одна з них довжиною 5, друга довжиною 2.

Приклади

Вхідні дані	Результат роботи
5 5 3 2	2
7 5 5 2	2

Ідея розв'язання задачі

Гаха С. О., учителя інформатики Ніжинської гімназії № 3 Ніжинської міської ради

Розв'язання:

```
var d:array[-4000..4000] of longint; n,a,b,c,i:longint;
f1,f2:text;
function max(x,y:longint):longint;
begin
 if x>y then max:=x else max:=y;
end;
begin
assign(f1,'ribbon.in');
assign(f2,'ribbon.out');
```

```
reset(f1);
rewrite(f2);
read(f1,n,a,b,c);
d[0]:=1;
for i:=1 to n do
 begin
  if d[i-a]<>0 then d[i]:=max(d[i],d[i-a]+1);
  if d[i-b]<>0 then d[i]:=max(d[i],d[i-b]+1);
  if d[i-c]<>0 then d[i]:=max(d[i],d[i-c]+1);
 end;
writeln(f2,d[n]-1); close(f2);
end.
```

Ідея розв'язання задачі

Бондарчук Н. І., учителя інформатики Добрянської загальноосвітньої школи І-ІІІ ст. Ріпкинського району

Розв'язання:

```
var n,a,b,c,k,i,j,x:longint;
begin
assign(input,'ribbon.in'); reset(input);
assign(output,'ribbon.out'); rewrite(output);
read(n,a,b,c);
k:=0;
for i:=0 to n div a do
  for j:=0 to (n-i*a) div b do
    begin
    if (n-i*a-j*b) mod c=0
      then begin
             x:=i+j+(n-i*a-j*b) div c;
             if x>k then k:=x;
            end;
    end;
writeln(k)
end.
```

Всього тестів: 56, пройдено 55, не пройдено 1.
Отримано балів: 98 (із 100).

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи I-III ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи I-III ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Ідея розв'язку: Визначення максимальної кількості частин довжинами a,b,c та пошук необхідної суми.

Розв'язання:

```
var i,j,k,n,b,c,x,y,z,s:longint;
    a:array[1..3] of longint; f:text;
    Ok:boolean;
Begin
 Assign(f,'ribbon.in');Reset(f);Read(f,n);
 For i:=1 to 3 do Read(f,a[i]); Close(f);
 For i:=1 to 2 do
 For j:=i to 3 do
 If a[i]>a[j] Then Begin x:=a[i]; a[i]:=a[j]; a[j]:=x;End;
 x:=n div a[1];{число частин довжини a}
 y:=n div a[2];{ число частин довжини b}
 z:=n div a[3];{ число частин довжини c}
 s:=0;
 Ok:=False;
{Потрійний цикл для набору заданої довжини стрічки, яка заноситься у величину S}
 For i:=0 to z do
  Begin
```

```
  For j:=0 to y do
   Begin
    For k:=x downto 0 do
     Begin
      b:=i*a[3]+j*a[2]+k*a[1];
      If n=b Then c:=i+j+k;
      If c>s Then s:=c;
      If s>x-2 Then Ok:=true;
      If Ok Then Break;
     End;
    If Ok Then Break;
   End;
  If Ok Then Break;
 End;
 Assign(f,'ribbon.out');Rewrite(f);WriteLN(f,s);Close(f);
End.
```

C – Заробітна плата

Ім'я вхідного файлу: salary.in

Ім'я вихідного файлу: salary.out

Обмеження часу: 300 мс

Обмеження пам'яті: 128 M

В деякій компанії працюють три співробітника – Олексій, Віктор і Сергій. Їх місячний оклад становить ***A, B, C*** грн відповідно. При цьому Олексій працює на повну ставку, а Віктор і Сергій – на половину ставки, тобто працюють вдвічі менше, ніж Олексій.

За підсумками місяця директор компанії хоче розподілити між цими співробітниками преміальний фонд, який складає ***N*** грн. При цьому директор хоче розподілити преміальний фонд таким чином, щоб підсумкова зарплата (сума окладу і премії) у цих співробітників виявилася пропорційна проведеним на роботі часу, тобто зарплата Олексія повинна виявитися рівно в два рази більше, ніж зарплата Віктора та Сергія. Більш формально, якщо

премія Олексія складе ***X*** грн, премія Віктора – ***Y*** грн, премія Сергія – ***I*** грн, то $A + x = 2(B + y) = 2(C + i)$, $x + y + i \le N$. При цьому бухгалтерія вимагає, щоб розмір премії (як і розмір окладу) була цілим числом грн, а директор хоче розподілити якомога більше преміального фонду, тобто сума $x + y + i$ повинна бути максимально можливою, не перевищуючи при цьому ***N***. Напишіть програму, яка визначить, яку премію потрібно призначити кожному з працівників.

Формат вхідних даних: Вхідний файл містить чотири цілих числа ***A, B, C,*** записані в окремих рядках, - розміри окладів Олексія, Віктора та Сергія $(A > 0, B > 0, C > 0)$. У четвертому рядку вхідних даних записано одне ціле число ***N*** – розмір преміального фонду $N \ge 0$).

Формат вихідних даних: Вихідний файл має містити три числа – розмір премії Олексія, Віктора та Сергія. Якщо преміальний фонд не можна розподілити так, щоб виконувалися необхідні умови, програма повинна вивести одне число 0.

Обмеження і система оцінювання: Рішення, яке виводить правильну відповідь тільки на тестах з умови і тих тестах, на яких відповіддю є «0», оцінюватиметься в 0 балів.

Рішення, правильно працююче у випадку, коли всі вхідні числа не перевищують 100, буде оцінюватися в 30 балів.

Рішення, правильно працююче у випадку, коли всі вхідні числа не перевищують 10^5, буде оцінюватися в 60 балів.

Рішення, правильно працююче у випадку, коли всі вхідні числа не перевищують 10^9, буде оцінюватися в 100 балів.

Пояснення до прикладів: Перший приклад: З урахуванням премії зарплата Олексія складе 12 грн, Віктора та Сергія – 6 грн. Другий приклад: Домогтися потрібного співвідношення преміальних виплат неможливо.

Приклади

Вхідні дані	**Результат роботи**
7 3 4	5 3 2

12	
20 10 11 2	0

Ідея розв'язання задачі

Бондарчук Н. І., учителя інформатики Добрянської загальноосвітньої школи I-III ст. Ріпкинського району

Розв'язання:

```
var a,b,c,n,ch:longint;
begin
assign(input,'salary.in'); reset(input);
assign(output,'salary.out'); rewrite(output);
readln(a);
readln(b);
readln(c);
readln(n);
ch:=(a+b+c+n) div 4;
if (a<=2*ch) and
   (b<=ch) and
   (c<=ch)
then begin
       writeln(2*ch-a);
       writeln(ch-b);
       writeln(ch-c);
      end
else writeln(0);
end.
```

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Скласти програму для розв'язання системи $\begin{cases} a+x=2(b+y) \\ a+x=2(c+z), \\ x+y+z \le n \end{cases}$ де *x, y, z* є невідомими величинами, причому сума *x+y+z* повинна бути найбільша та не перевищувати преміювальний фонд *n*.

Розв'язання:

```
 Type massiv=array[1..3]  of longint;
 var i,j,n,a,b,c:longint; x,y,z:int64; f:text; d:massiv;
     Ok:boolean;
Begin
 Assign(f,'salary.in');Reset(f);
 Readln(f,a);Readln(f,b);Readln(f,c);Readln(f,n);Close(f);
 j:=(n-a+b+c) div 2 ;
 For i:=j downto 0 do
  Begin
   ok:=((a+i-2*b) mod 2=0) and ((a+i-2*c) mod 2=0) and
     (a+i-2*b>=0) and (a+i-2*c>=0);
   If ok Then Begin x:=i; y:=(a+x-2*b) div 2;
                      z:=(a+x-2*c) div 2; Break;
               End;
   If (a+i-2*b<0) or (a+i-2*c<0) Then Break;
```

```
  End;
 Assign(f,'salary.out');Rewrite(f);
 If Ok Then Begin WriteLN(f,x);WriteLN(f,y);WriteLN(f,z);End
      Else WriteLN(f,x);
 Close(f);
End.
```

D – Поближче до буфету

Ім'я вхідного файлу: buffet.in

Ім'я вихідного файлу: buffet.out

Обмеження часу: 500 мс

Обмеження пам'яті: 128 M

Степан нещодавно був студентом, а вже декан. Зараз він навіть розклад складає для своєї улюбленої групи. Всім відомо, що після лекції студенти першим ділом біжать в буфет, для цього він хоче знайти таку лекційну аудиторію, з якої можна найшвидше добратись до буфету. На його факультеті N аудиторій, між аудиторіям є швидкі доріжки, по яких можна швидко пересуватись від аудиторії до аудиторії в обох напрямках, рух по кожній з яких займає якусь кількість секунд. В деяких аудиторіях знаходяться буфети, деякі аудиторії – лекційні.

Формат вхідних даних: В першому рядку знаходяться два числа N і M ($2 \le N \le 5000$, $1 \le M \le 100000$) – кількість аудиторій і кількість швидкісних доріжок відповідно.

В другому рядку знаходиться N цілих чисел a_i ($0 \le a_i \le 2$). Якщо $a_i = 0$, то в цій аудиторії не буфет, а також вона не є лекційною. Якщо $a_i = 1$, то ця аудиторія є лекційною. Якщо $a_i = 2$, то в цій аудиторі знаходиться буфет. Гарантується, що хоча б одне з цих чисел дорівнює одиниці, і хоча б одне дорівнює двійці.

В кожному з наступних ***M*** рядків знаходиться по три числа x_i y_i z_i, які позначають, що аудиторії x_i і y_i ($1 \le x_i, y_i \le n, x_i \ne y_i$) з'єднані швидкісною доріжкою, час руху по якій дорівнює z_i секунд. Гарантується, що дві аудиторії не з'єднані більш ніж одною доріжкою.

Формат вихідних даних: Якщо існує такий шлях, по якому можна добратись з лекційної до буфету якнайшвидше, то виведіть три числа: ***x, y, d***, які позначають, що з лекційної ***x*** є шлях до аудиторії ***y*** з сумарним часом ***d*** хвилин. Аудиторія ***x*** має бути лекційною, в аудиторії ***y*** має бути буфет. Якщо існує декілька відповідей, виведіть будь-яку з них. Якщо такого шляху не існує, виведіть ***-1***.

Приклади

Вхідні дані	Результат роботи
6 6 1 1 0 0 2 2 1 3 3 1 2 3 1 5 6 3 6 2 2 3 4 2 4 5	1 6 5

Ідея розв'язання задачі

Гаха С. О., учителя інформатики Ніжинської гімназії № 3 Ніжинської міської ради

За допомогою алгоритму Флойда знайдемо найкоротші шляхи між будь якою парою аудиторій. Потім переберемо всі пари, і якщо перша вершина це лекційна, а в другій - буфет, і відстань між цими вершинами найменша, то відповіддю і буде шлях між цими вершинами.

Розв'язання:

```
var a:array[1..5000,1..5000] of longint;
 b:array[1..5000] of longint;
i,j,k,m,n,min,x,y,z:longint;
f1,f2:text;
begin
assign(f1,' buffet.in');
assign(f2, buffet.out');
reset(f1);
rewrite(f2);
read(f1,n,m);
for i:=1 to n do
for j:=1 to n do
a[i,j]:=maxlongint div 2;
for i:=1 to n do
read(f1,b[i]);
for i:=1 to m do
begin
read(f1,x,y,z);
a[x,y]:=z;
a[y,x]:=z;
end;
for k:=1 to n do
 for i:=1 to n do
 for j:=1 to n do
   if a[i,k]+a[k,j]<a[i,j] then
a[i,j]:=a[i,k]+a[k,j]; min:=maxlongint;
 for i:=1 to n do
  for j:=1 to n do
   if (a[i,j]<min) and (b[i]=1) and (b[j]=2) then
       begin
       min:=a[i,j];
       x:=i;
       y:=j;
       end;
```

```
if min<maxlongint div 2 then writeln(f2,x,' ',y,' ',min) else writeln(f2,-1); close(f2);
end.
```

Ідея розв’язання задачі

Зеленського О.С., учителя інформатики та фізики, учителя-методиста Куликівської загальноосвітньої школи І-ІІІ ст. Куликівської районної ради

Створимо дві фіктивні вершини, назвемо їх S і T. Поєднаємо всі вершини, які позначають лекційні з вершиною S ребрами вартості 0. А також поєднаємо всі вершини, які позначають лекції з буфетом, з вершиною T, також вартості 0. Потім за допомогою алгоритму Дейкстри (бажано з допомогою кучі, щоб складність пошуку найкоротшого шляху була не O(N2), а O(MlogN)) знаходимо шлях від вершини S до вершини T, це і буде оптимальним шляхом, за виключенням першої і останньої вершини.

Складність O(M logN)

Так як граф отримуємо розріджений (кількість ребер значно менша квадрату кількості вершин), то представимо граф у вигляді зв'язаного списку, розміщеного в динамічній пам'яті.

Розв’язання:

```
const
  maxn = 5002;
  maxm = 25020004;
  inf  = 223372036854775807;

type
  pnode = ^tnode;
  tnode = record
    v: int64;
    weight: int64;
    next: pnode;
  end;
```

```
var
  a: array [1..maxm] of pnode;
  d: array [0..maxn] of int64;
  p: array [1..maxn] of int64;
  route:array[0..maxn] of int64;
  visited: array [1..maxn] of boolean;
  s: int64;
  n, M: int64;
  aa:array[1..maxn] of 0..2;
  start,fin,x:int64;
    i,j: int64;

//Процедура додавання вершини до списку
procedure insert_vertex(v, w: int64; weight: int64);
var
  p: pnode;
begin
  new(p);
  p^.v := w;
  p^.weight := weight;

  p^.next := a[v];
  a[v] := p;
end;


procedure init;
var
  i, j, x, y, nn, z: int64;
begin

 for i := 1 to maxn do
  begin
    a[i] := nil;
    d[i] := inf;
  end;

  fillchar(p, sizeof(p), 0);
```

```
  fillchar(visited, sizeof(visited), false);


        readln(N,M);
        For i := 1 To N   do begin
                Read (aa[i]);
                //Якщо a = 1, то ця аудиторія є лекційною.
                //Якщо a = 2, то в цій аудиторії знаходиться буфет.
        end;
        //For i := 1 To N do write(aa[i],' '); writeln;
        Readln;
        for i := 1 to m do begin
           read(x, y, z);
           insert_vertex(x, y, z);
           insert_vertex(y, x, z);
        end;
        //Створимо дві віртуальні вершини, перша з яких зв'язана однонапрямленими ребрами нульової ваги з усіма лекційними аудиторіями, а з другою з'єднані аналогічними ребрами усі аудиторії з буфетами.
        n:=n+2;
        start:=n;
        fin:=n-1;
        For i := 1 To n-2   do begin
                if aa[i]=1 then begin
                        insert_vertex(start, i, 0);
                end;
                if aa[i]=2 then begin
                        insert_vertex(i, fin, 0);
                end;
        end;

end;


procedure dijkstra(s: int64);
var
```

```
  j, i, min, v, weight: int64;
  t: pnode;
begin
  t := a[s];
  while t <> nil do
  begin
    d[t^.v] := t^.weight;
    t := t^.next;
  end;

  for v := 1 to n do
    p[v] := s;
  d[s] := 0;
  p[s] := 0;
  visited[s] := true;

  for j := 2 to n do
  begin
    min := inf;
    for i := 1 to n do
      if (not visited[i]) and (d[i] < min) then
      begin
        min := d[i];
        v := i;
      end;
    t := a[v];
    visited[v] := true;
    while t <> nil do
    begin
      if d[t^.v] > d[v] + t^.weight then
      begin
        d[t^.v] := d[v] + t^.weight;
        p[t^.v] := v;
      end;
      t := t^.next;
    end;
  end;
```

```
end;

procedure print_path(x: int64);
begin

        if x <> s then
   print_path(p[x]);
        write(x, ' ');

end;


begin
        assign(input,'buffet.in');
        assign(output,'buffet.out');
        reset(input);
        rewrite(output);
        init;
        //За допомогою алгоритма Дейкстри знайдемо шляхи від першої (віртуальної) вершини, з'єднаної ребрами нульової ваги з усіма лекційними аудиторіями
        dijkstra(start);
        x:=fin; i:=1;
        //В масив  route запишемо послідовність пройдених найкоротлим шляхом ребер
        while x <> start do begin
                route[i]:=x;
                x:=p[x];
                inc(i);
        end;
        route[i]:=x;
   j:=route[i];
        //знайдемо вершину, наступну за фіктивною початковою
   while j<>start do begin
                dec(i);
                j:=route[i];
   end;
```

```
  dec(i);
        //Цю вершину (i) і використаємо як номер початкової
        if d[fin]<>inf then
                writeln(route[i],' ',route[2],' ',d[fin])
        else
                writeln(-1);
//Звільнимо динамічну пам'ять
        for i := 1 to n do dispose(a[i]);
        close(input);
        close(output);
end.
```

Ідеї розв’язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв’язання програм написані в середовищі Free Pascal 2.6.4.

Класична задача про граф із застосуванням алгоритму Дейкстри (пошук найкоротшого шляху з аудиторії типу 1 (лекційної) у буфет).

Розв’язання:

```
Type massiv=array[0..5001,0..5001]  of longint;
mas=array[0..5001] of longint;
var i,j,n,m:longint; x,y,z:int64; f:text; a:massiv; c,d:mas;
   visit:array[0..5001] of boolean;

Function Deykstra(aa,bb:longint):longint;
 Var v,min,u,s:longint;
Begin
```

```
  c[aa]:=0; visit[aa]:=true;d[aa]:=0;
  For v:=0 to n do
   Begin
    If d[n+1]<>1000000000 Then Break;
    u:=0; min:=1000000000;
    For s:=1 to n+1 do
     If (not visit[s]) and (d[s]<min)
      Then Begin min:=d[s];u:=s; End;
    visit[u]:=true;
    For s:=1 to n+1 do
    If (not visit[s]) and (d[s]>d[u]+a[u,s]) Then
     Begin d[s]:= d[u]+a[u,s]; c[s]:=u; End;
   End;
  Deykstra:=d[n+1];
 End;

Begin
 Assign(f,'buffet.in');Reset(f);Readln(f,n,m);
 For i:=0 to n+1 do
  Begin
   For j:=0 to n+1 do a[i,j]:=1000000000;
   d[i]:=a[0,i];
   visit[i]:=false;
  End;

 For i:=1 to n do
  Begin
   Read(f,x);
   If x=1 Then Begin a[i,0]:=0;a[0,i]:=0;End;
   If x=2 Then Begin a[i,n+1]:=0;a[n+1,i]:=0;End;
  End;Readln(f);
 For i:=1 to m do
  Begin ReadLN(f,x,y,z);a[x,y]:=z; a[y,x]:=z;End;
 Close(f);

 x:=Deykstra(0,n+1); z:=n+1;
 For i:=1 to n+1 do
```

```
  Begin z:=c[z]; If z=0 Then Break; y:=z;End;
 Assign(f,'buffet.out');Rewrite(f);
 If x<>1000000000 Then WriteLN(f,y,' ',c[n+1],' ',x)
    Else WriteLN(f,-1);   Close(f);
End.
```

E – Многокутник

Ім'я вхідного файлу: polygon.in

Ім'я вихідного файлу: polygon.out

Обмеження часу: 300 мс

Обмеження пам'яті: 128 M

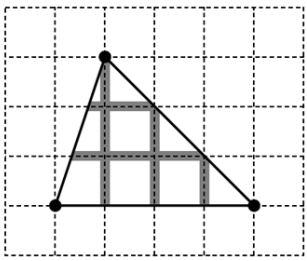
Площину покрили перпендикулярними прямими виду ***x=a*** та ***y=b*** де ***a*** та ***b*** – цілі числа. В результаті утворилась сітка. Многокутник складається з ***N*** вершин, що мають цілочисельні координати. Потрібно знайти сумарну довжину відрізків сітки, що лежать строго в даному многокутнику (якщо відрізок сітки співпадає з ребром многокутника – рахувати не треба). Для наочності можна скористатись малюнком.

Вхідні дані: В першому рядку міститься число ***N (3 ≤ N ≤ 100000).*** В кожному із наступних ***N*** рядків містяться два цілих числа – координати вершини многокутника ***(-5·10^8 ≤x, y ≤ 5·10^8).*** Координати задаються в порядку обходу або за, або проти годинникової стрілки.

Вихідні дані: Вивести одне єдине число (з точністю до 3 знаків після десяткової крапки) – сумарну довжину відрізків сітки, що лежать строго в даному многокутнику.

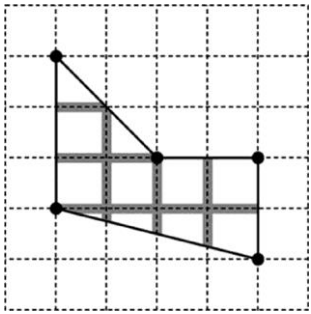
Оцінювання: В даній задачі кожен тест оцінюється окремо.

Пояснення прикладів:



Перший тест.

Сумарна довжина горизонтальних ліній: ***4/3 + 8/3 = 4.***



Сумарна довжина вертикальних ліній: ***3 + 2 + 1 = 6***. Тому загальна довжина рівна ***4 + 6 = 10***
Другий тест.

Сумарна довжина горизонтальних ліній: ***1+2+4 = 7***.
Сумарна довжина вертикальних ліній: ***9/4+3/2+7/4 = 5.5***. Тому загальна довжина рівна ***7 + 5.5 = 12.5***.

Приклади

Вхідні дані	Результат роботи
3 5 1 2 4 1 1	10.0
5 0 0 -2 2 -2 -1 2 -2 2 0	12.5

Ідея розв'язання задачі

Гаха С. О., учителя інформатики Ніжинської гімназії № 3 Ніжинської міської ради

Розв'язання:

```
var x,y:array[0..100000] of int64; i,n:longint;
s,l1,l2:int64;
```

```
f1,f2:text; begin
assign(f1,'poligon.in');
assign(f2,'poligon.out');
reset(f1);
rewrite(f2);
read(n);
for i:=1 to n do
read(f1,x[i],y[i]);
x[0]:=x[n];y[0]:=y[n];
for i:=1 to n do begin
  if y[i]=y[i-1] then l1:=l1+abs(x[i]-x[i-1]); if x[i]=x[i-1] then
l2:=l2+abs(y[i]-y[i-1]); end;
for i:=1 to n do s:=s+(x[i]-x[0])*(y[i-1]-y[0])-(y[i]-y[0])*(x[i-1]-
x[0]); writeln(f2,abs(s)-(l1+l2)/2:0:1);
close(f2);
end.
```

Ідея розв'язання задачі

Зуба В. В., учителя математики, директора Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста;

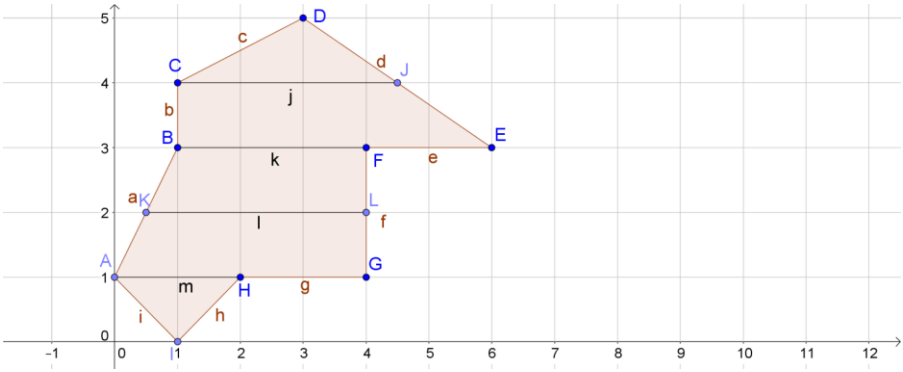
Бондаренка С. М., учителя математики та інформатики Прилуцької загальноосвітньої школи І-ІІІ ступенів № 7 Прилуцької міської ради, учителя-методиста

Розв'язання програм написані в середовищі Free Pascal 2.6.4.

Для розв'язання задачі застосовуємо формулу обчислення площі n–кутника: $\mathbf{0.5}\sum_{i=1}^{n+1}(x_i \cdot y_{i+1} - x_{i+1} \cdot y_i)$, де ***n+1=1***.

Також площа n–кутника дорівнює сумі площ трапецій висотою 1, де площа трапеції обчислюється за формулою $S = \frac{a+b}{2} \cdot h$. Тоді шукана сума горизонтальних та вертикальних відрізків буде сумою основ цих трапецій, причому кожна

сторона враховується два рази, крім вертикальних та горизонтальних сторін многокутника.



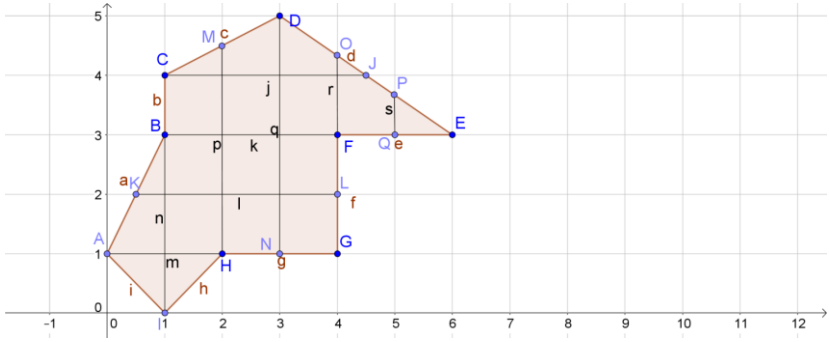
Наприклад, площа многокутника зображеного на рис.1 дорівнює сумі площ 2 трикутників та 3 трапецій. Висота цих п'яти фігур ***h=1***.

Сумарну довжину горизонтальних відрізків сітки (***j,k,e,i,m,g)***, що лежать строго в даному многокутнику знаходимо таким чином:

s=0.5*(jh+(j+k+e)h+(k+i)h+(i+m+g)h+mh)= .5*(j+j+k+e+k+i+i+m+g+m)= =0.5*(2j+2k+e+2i+2m+g).

Горизонтальні основи враховуються лише один раз.

Обчислення сумарної довжини вертикальних відрізків сітки: ***s=0.5*(2s+2r+f+2q+2p+2n+b).***



Розв'язання:

```
var i,j,n:longint; x,y,x1,x2,y1,y2,a,b,s:int64;
Begin
  Assign(input,'polygon.in');Reset(input);Readln(n);
  Assign(output,'polygon.out');Rewrite(output);
  Readln(x,y);x1:=x; y1:=y;
  For i:=1 to n-1 do
   Begin
    Readln(x2,y2);
    s:=s+x1*y2-x2*y1;{Обчислення площі многокутника}
    If x1=x2 Then a:=a+abs(y2-y1);{Обчислення сумарної довжини
горизонтальних сторін многокутника}
    If y1=y2 Then b:=b+abs(x2-x1);{ Обчислення сумарної
довжини вертикальних сторін многокутника }
    x1:=x2; y1:=y2;
   End;
{Остання вершина многокутника поєднується з першою}
  x2:=x; y2:=y;
  s:=s+x1*y2-x2*y1;
  If x1=x2 Then a:=a+abs(y2-y1);
  If y1=y2 Then b:=b+abs(x2-x1);
  Close(input);
  WriteLN(0.5*(2*abs(s)-a-b):0:3);
  Close(output);
End.
```