Ресурси мережі Internet

http://acmicpc.org	Офіційний сайт ACM-ICPC
http://e-olimp.com.ua	Система підготовки та проведення олімпіад зі спортивного програмування
http://acm.lviv.ua	ACM-Contester Український портал ACM-спільноти
http://acm.mipt.ru	Олімпіади з програмування на Фізтехі
http://acm.timus.ru	Timus Online Judge — найкрупніший в Росії архів задач з різноманітних змагань зі спортивного програмування.
http://acmsolver.org	Портал для тих кого цікавлять змагання з програмування ACM-ICPC
http://g6prog.narod.ru	Розбір олімпіадних задач з інформатики від Михайла Густокашина
http://ips.ifmo.ru	Російська Інтернет-школа інформатики та програмування
http://informatik.kz	Інформатика викладання та навчання

Олімпіадні задачі з *інформатики*

В.Є. Величко, М.М. Рубан,
В.П. Батуніна, С.Є. Устінов

Розв'язання задач
II етапу Всеукраїнської олімпіади
з інформатики – 2007, 2008 рр.

Слов'янськ 2009

Серія заснована у 2008 році

ББК 32.973-018
УДК 374.1:004.021

В.Є. Величко, М.М. Рубан, В.П. Батуніна, С.Є. Устінов ОЛІМПІАДНІ ЗАДАЧІ З ІНФОРМАТИКИ: Розв'язання задач II етапу Всеукраїнської олімпіади з інформатики – 2007, 2008 рр.. – Слов'янськ, 2009. – 34 с.

Для вчителів, методистів, керівників гуртків з програмування, викладачів та студентів спеціальностей "математика "інформатика" вищих навчальних закладів.

Рецензенти: кандидат фіз-мат наук, с.н.с. САПУНОВ С.В., Інститут прикладної математики та механіки НАН України, лабораторія дискретної математики та прикладної алгебри

кандидат пед. наук, доцент СЬОМКІН В.С., Слов'янський державний педагогічний університет, доцент кафедри ГМВМ

Схвалено Радою фізико-математичного факультету СДПУ. Протокол №4 від 24 грудня 2008 року

Затверджено Вченою радою Слов'янського державного педагогічного університету. Протокол №5 від 30 грудня 2008 року

Відповідальний за випуск кандидат фіз.-мат. наук, завідувач кафедри алгебри Величко В.Є.

Література

[1] **Ахо А., Хопкрофт Дж., Ульман Дж.** *Структуры данных и алгоритмы.* – Москва: Вильямс, 2007. — 400с.

[2] **Беллман Р., Дрейфус С.** *Прикладные задачи динамического программирования.* – Москва: Наука, 1965. — 458с.

[3] **Грэхем Р., Кнут Д., Паташник О.** *Конкретная математика.Основы информатики.* М: Мир, 2006. – 704с.

[4] **Долинский М.С.** *Алгоритмизация и программирование на Turbo Pascal от простых до олимпиадных задач.* СПб.: Питер, 2005. – 237с.

[5] **Иванов Б.Н.** *Дискретная математика. Алгоритмы и программы.* М.:Лаборатория Базовых Знаний, 2001. – 288с.

[6] **Кнут Д.** *Искусство программирования, в 3-х томах.* М.: Вильямс, 2000.

[7] **Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К.** *Алгоритмы: построение и анализ.* М.: Вильямс, 2005. – 1296с.

[8] **Меньшиков Ф.** *Олимпиадные задачи по программированию.* – СПб.: Питер, 2005.— 320с.

[9] **Порублев И.Н., Ставровский А.Б.** *Алгоритмы и программы. Решение олимпиадных задач.* – Москва: Вильямс, 2007.— 480с.

[10] **Радион В.С.** *Олимпиады по информатике: задачи, решения, тесты.* – Минск: Аверсэв, 2007.— 367с.

[11] **Седжвик Р.** *Фундаментальне алгоритмы на С.* Киев: Диасофт, 2001. – 688с.

[12] **Скиена С. С, Ревилла М. А.** *Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям*/ Пер. с англ. – М: КУДИЦ-ОБРАЗ, 2005. — 416с.

[13] **Шень А.** *Программирование: теоремы и задачи.* — 2-е изд., испр. и доп. — М.: МЦНМО, 2004. — 296с.

Задача *D*. Пригоди Кролика.

Одного разу під час прогулянки лісом Кролик знайшов печеру, в якій була кімната з магічним малюнком на підлозі, малюнок складається із плит двох видів – правильних 8-кутників та 4-кутників. Кімната має форму квадрата розміром $N \times N$ правильних 8-кутників (приклад на малюнку). Як виявилося, пересуватися по кімнаті можна тільки за певними правилами. З кожної 8-кутної плити підлоги (крім тих, які примикають до відповідних стін кімнати) можна перейти або на сусідню праву плиту, або на сусідню верхню, або по діагоналі на сусідню 4-кутну плиту. З кожної 4-кутної плити можна перейти на 8-кутну плиту, що примикає праворуч згори або знизу. Всі плити послідовно построково занумеровані починаючи з лівої нижньої плити.

На плиті з номером K стоїть скриня з морквою - кращого скарбу для Кролика не знайти. Вихід з печери розташований в куті, протилежному до початкового. Допоможіть Кролику дізнатися скількома різними способами він зможе добратися до скрині, користуючись правилами руху в кімнаті, а також вибратися з магічної кімнати, враховуючи, що на першу плиту можна потрапити одним способом.

Вхідні дані. В єдиному рядку записані два цілих числа N та K ($1 \le N \le 100; K \le N$).

Вихідні дані. Два числа через пробіл, перше - кількість можливих способів перейти з першої плити на останню плиту в кімнаті; друге - кількість способів добратися з першої плити до плити на якій розташована скриня з морквою.

Приклад вхідних і вихідних даних введення виведення.

Приклад введення:	*Приклад виведення:*
2 2	4 2
3 7	28 5

Зміст

Вступ

Олімпійський рух з інформатики набув великої популярності в наш час як в усьому світі, так і в нашій країні: проводяться шкільні, міські, районні, обласні, державні і міжнародні олімпіади та турніри. З року в рік росте кількість учнів та студентів, які вивчають інформатику на поглибленому рівні і бажаючих вивчати її з більш раннього шкільного віку.

Чудовими матеріалами для підготовки до нових олімпіад є збірники олімпіадних задач які вже проходили. І хоча з інформатики проведено вже велику кількість олімпіад та турнірів, а з задачами, які там було запропоновано можна ознайомитись на сайтах Інтернету, систематизованих і методично оформлених збірок в навчальних закладах практично не існує.

Даний випуск містить задачі олімпіад II етапу, які проходили в Донецькій області в 2007 і 2008 роках. І хоча для 8-9 класів умовою задачі не передбачалось написання комп'ютерної програми, але наведені приклади розв'язання задач будуть дуже корисними як для учнів 10-11 класів так і для студентів спеціальностей "математика" та "інформатика".

До деяких задач ми використовували більш жорсткі умови розв'язання. Ці умови не ставились як обов'язкові під час проведення II етапу, але вони завжди присутні при проведенні міжнародних олімпіад та турнірів, що проходять під егідою ACM (Association for Computing Machinery).

Основна мета посібника – допомога:

- вчителю, викладачу, тренеру студентів – системно організувати підготовку учнів, до олімпіад і турнірів з інформатики;

- учням, студентам – перевірити рівень своєї підготовки, розв'язуючи завдання із збірника і оцінюючи свої розв'язки;

- всім учасникам олімпійського руху з інформатики – підвищити свій олімпійський потенціал, ознайомившись із запропонованими розв'язками задач.

Як краще підготуватися до олімпіади. Будь яка олімпіада виставляє більш високі вимоги до знань, умінь та навичок, ніж ті, які необхідні на звичайних шкільних уроках. А тому до олімпіади треба готуватись додатково.

Учасник олімпіади з інформатики (програмування) повинен:

- досконало володіти навичками користувача персонального комп'ютера;

Задача *C*. Дорога додому.

Повертаючись, додому, після захоплюючої гри в гостях у Вінні Пуха, ослик Іа вирішив трохи прогулятися. Оскільки під час прогулянки він увесь час думав про свій день народження, який наближався, то не помітив, як заблукав. Відомо, що ослик під час прогулянки завжди пересувається за певним алгоритмом: на початку прогулянки він завжди починає рух на північний схід, проходячи при цьому один крок (переміщаючись при цьому в точку (1,1)), потім міняє напрямок і рухається на південний схід, далі на південний захід, та на північний захід і так далі. При кожній зміні напрямку ослик завжди робить на N-кроків більше, ніж було зроблено до зміни напрямку.

Коли ослик все ж таки вирішив повернутися додому, то виявив що зайшов глибоко у ліс. Оскільки насувалася ніч, Іа захотів якнайшвидше потрапити додому. Допоможіть дізнатися, чи вдасться сьогодні осликові потрапити додому до заходу сонця, якщо відомо, що сонце зайде через T-годин, а швидкість пересування ослика V-кроків на годину, враховуючи те, що довжина кроку у ослика стала. Відомо, що рух ослик почав з точки з координатами (0, 0), а його будинок розташований у точці (h_x, h_y), та напрямок руху він міняв K-разів.

Вхідні дані. У першому рядку дано чотири цілих числа N, K, T, V ($0 \le N, K, T, V \le 100$). У другому рядку розташовано два цілих числа h_x, h_y - координати будинку ослика ($-10^5 \le h_x, h_y \le 10^5$).

Вихідні дані. Вивести "Good night Ia" (без лапок), якщо ослик встигне дійти додому до заходу сонця та "Poor Ia" в іншому випадку.

Приклад вхідних і вихідних даних введення виведення.

Приклад введення:	*Приклад виведення:*
1 2 3 4 3 0	Good night Ia
4 3 2 1 0 3	Poor Ia

Задача *B*. Друзі Вінні.

Після успішної будівлі паркану, Вінні Пух вирішив трохи відпочити та пограти зі своїми друзями в одну цікаву гру.

Вінні взяв камінці і розклав їх у ряд в M куп. Кожен з друзів Вінні підходив і брав саму ліву купу камінців та розкладав всі взяті камінці в усі наступні купи по одному камінцю. Якщо куп камінців менше, ніж камінців взятих другом, то камінці які залишилися утворюють нові купи, так триває поки не походить останній друг Вінні. Після гри на полі залишається N куп камінців.

Приклад такої гри:

Початковий стан	7 5 1 3 6
Друг #1	6 2 4 7 1 1 1
Друг #2	3 5 8 2 2 2
Друг #3	6 9 3 2 2

Ваше завдання, знаючи початкове розташування камінців по купах і кінцевий стан, визначити кількість друзів Вінні.

Вхідні дані. У першому рядку дано два числа - M та N ($2 \leq M, N \leq 1000$), M - кількість куп камінців у початковому стані, N - кількість куп камінців після гри. У другому рядку знаходяться M цілих чисел (початковий стан), M_i - кількість камінців у i-ій купі ($1 \leq M_i \leq 100$). У третьому рядку відповідно знаходяться N цілих чисел ($1 \leq N_i \leq 100$).

Вихідні дані. Вивести одне число - кількість друзів Вінні.

Приклад вхідних і вихідних даних введення виведення.

Приклад введення:	*Приклад виведення:*
6 5 7 4 3 2 1 9 5 13 4 3 1	4

- уміти створювати, редагувати і налагоджувати програми в одній із відомих мов програмування високого рівня;

- знати типові алгоритми і основні методи алгоритмізації і вміти їх використовувати при розв'язуванні задач;

- мати досвід розв'язання олімпіадних задач з інформатики.

На основі вищенаведених вимог, кожен може скласти собі програму підготовки до олімпіади, де одним з основних пунктів повинен бути розв'язок задач, що були запропоновані на попередніх олімпіадах.

Ми використовуємо мову *Pascal* для представлення алгоритмів які описуємо, тому, що це одна з найпоширеніших мов програмування. Спочатку алгоритми представляються у вигляді аналізу чи в іншій абстрактній формі. Це зроблено для того, щоб показати весь спектр проблем при розв'язку практичних задач: від проблеми формалізації задачі до проблем, які можуть виникнути під час виконання завершеної програми.

В тексті можна зустріти такі, не характерні для звичайної математики, оператори як *div* та *mod*, це, відповідно, цілочисленне ділення та операція знаходження залишку від ділення (наприклад, $17\ div\ 7 = 2$; $17\ mod\ 7 = 3$). Алгоритми, які ми пропонуємо, можна реалізувати на будь-якій відомій Вам мові програмування.

Як відомо, більшість олімпіадних задач передбачає використання файлів для введення та виведення даних. Проте часто зустрічається введення та виведення через стандартні потоки вводу-виводу (консоль). Тому під час розв'язку задач 8-9 класів ми використовували введення даних через консоль (так як в умові не було вказано формат введення/виведення даних), а для задач 10-11 класів введення та виведення даних проводиться за допомогою файлів (бо це вказано в умові).

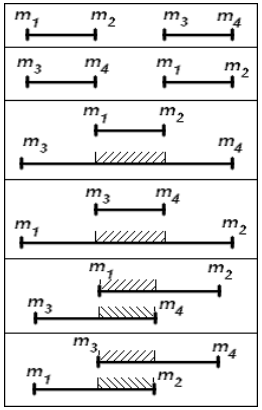
Бажаємо успіхів!

2007 рік

Все, чему мы хотим научиться, мы учимся, делая.
/Аристотель (Aristotle) Этика Никомахи II (325 до н.э.)/
Кто читает, тот совершенствуется.
/Хабаккук (Habakkuk) 2:2 (600 до н.э.)/
Кто совершенствуется, тот читает.
/Вильям Купер (William Cowper), Тиросиниум (1785)/

8-9 клас

Задача 1. Допит свідків. У справі про пограбування банку були допитані 2 свідка. Перший заявив, що пограбування відбулося десь між t_1 й t_2 годинами одного дня, а другий – що між t_3 й t_4 годинами того ж дня.



Складіть алгоритм, що визначає чи можуть обоє свідка одночасно говорити правду. Слово «між» позначає, що відповідний момент часу перебуває в межах від одного з них до іншого, але, не включаючи самі зазначені моменти часу. Наприклад, між 16 й 13 годинами лежать моменти часу 13:40, 15:00, але не 13:00, 16:50 або 8:15.

Зауваження. Оскільки в умові явно не вказано формат вхідних даних, будемо вважати, що час задається двома числами (години, хвилини).

Аналіз. Для позитивної відповіді в задачі необхідно розглянути, чи може бути в інтервалах (t_1, t_2) та (t_3, t_4) спільні точки. Оскільки час заданий кількістю годин та хвилин пропонується спочатку перевести його в хвилини, тобто ми отримаємо два інтервали цілих чисел (m_1, m_2) та (m_3, m_4). Так як нам невідомо, чи впорядковано було названо границі пограбування, то попередньо відсортуємо їх за зростанням, а потім перевіримо їх взаємне розташування: якщо інтервали не мають жодної спільної точки, то хтось зі свідків бреше, інакше вони кажуть правду.

```
1 var t1h,t1m,t2h,t2m,t3h,t3m,t4h,t4m:byte;
2       m1,m2,m3,m4,v:integer;
3 begin
4  writeln('First interval');
5    readln(t1h,t1m,t2h,t2m);
6  writeln('Second interval');
7    readln(t3h,t3m,t4h,t4m);
8  m1:=t1h*60+t1m; m2:=t2h*60+t2m;
9  m3:=t3h*60+t3m; m4:=t4h*60+t4m;
```


Задачі для самостійного розв'язку

Задача *A*. Охорона меду.

У ведмедя Вінні Пуха є своя пасіка. Побоюючись того, що пасіку почнуть розоряти інші ведмеді, Вінні вирішив побудувати паркан навколо її. Однак коштів Вінні має обмаль, тому він вирішив зробити паркан мінімальної довжини, але вулики повинні бути на відстані від паркану не менш ніж на K метрів, щоб "довголапі"ведмеді не змогли дістати до них. Через любов до геометрії Вінні вирішив, що паркан повинен мати форму опуклого багатокутника, сторони якого паралельні осям координат. Крім цього Вінні стало цікаво, яка ж буде площа його пасіки після будівлі паркану. Допоможіть Вінні знайти довжину його майбутнього паркану, а також площу пасіки.

Вхідні дані. У першому рядку два числа розділені пробілом - N ($2 \leq N \leq 100\ 000$) і K ($0 \leq K \leq 100$) - кількість вуликів та відстань до паркану. У наступних N рядках знаходяться описи вуликів, кожен складається із двох цілих чисел X_i, Y_i ($|X_i| <= 10^9, |Y_i| <= 10^9$) - координати положення i-го вулика. Всі вулики розташовані в різних точках.

Вихідні дані. Вивести два числа через пробіл з точністю до трьох знаків після коми - мінімальну довжину паркану та площу пасіки.

Приклад вхідних і вихідних даних введення виведення.

Приклад введення:	*Приклад виведення:*
4 0 1 1 1 -1 -1 1 -1 -1	8.000 4.000

```
    begin
      if mas[k+k]=0 then mas[k]:=2;
    end
    else
    begin
      for j:=1 to 1000 do
        if (mas[j]=1) and (mas[k+j]=0) then mas[k+j]:=2;
      mas[k]:=1;
    end;
  end;
  readln;
  readln(m);
  for i:=1 to m-1 do
  begin
    read(k);
    write(ord((k<2001) and (mas[k]>0)),' ');
  end;
  readln(k);
  writeln(ord((k<2001) and (mas[k]>0)));
end.
```



```
  if m1>m2 then begin v:=m1; m1:=m2; m2:=v; end;
  if m3>m4 then begin v:=m3; m3:=m4; m4:=v; end;
  if (m2<m3) or (m1>m4)
    then writeln('False')
    else writeln('True');
end.
```

Задача 2. Округлення. Задано дійсне число x. Напишіть алгоритм округлення цього числа, тобто алгоритм, що знайде найближче до x ціле число. В алгоритмі дозволяється використати найпростіші арифметичні операції (додавання, віднімання, множення, ділення, порівняння чисел), але заборонено операції округлення, відсікання дробової частини й т.п. (всі, які дозволяють перетворити дійсне число в ціле, у рядок або привести ще до якого-небудь типу).

Аналіз. Спочатку знайдемо натуральне число, яке буде не менше модуля заданого дійсного числа. Наступним кроком перевіримо, чи буде знайдене число ближчим в порівнянні з попереднім натуральним числом. Для цього порівняємо відстані від заданого дійсного числа до знайденого натурального та натурального числа, що на одиницю менше. Останнім кроком буде визначення знака дійсного числа, і в разі коли воно від'ємне, замінимо знак у знайденого натурального числа.

```
var n:real;
    x:integer;
begin
  write('Input the real number');
  readln(n);
  x:=0;
  repeat   x:=x+1  until abs(n)<x;
  if x-abs(n)>abs(n)-x+1 then x:=x-1;
  if n<0 then x:=-x;
  writeln('A rounded value ',x);
end.
```

Задача 3. Новорічні подарунки. У Діда Мороза є N коробок цукерок. Кількість цукерок у кожній коробці задано в таблиці $A[1..N]$. X дітей хочуть одержати парне число цукерок, а Y дітей – непарне. Складіть алгоритм, що визначає, чи може Дід Мороз роздати всі N коробок цукерок дітям так, щоб X дітей одержали парне число цукерок, Y – непарне, і кожній дитині дісталась як мінімум одна коробка. Ділити коробки не дозволяється.

Аналіз. На початку розв'язку потрібно визначити кількість коробок з парним та непарним числом цукерок та непарне число цукерок. Позначимо ці величини K_X та K_Y відповідно. Алгоритм дій представимо у

вигляді таблиці.

Якщо	*Відповідні дії*	
$X+Y>N$	Оскільки кількість дітей більша за кількість коробок, зрозуміло, що не вийде задовольнити потреби кожної дитини.	
$X+Y=N$	$K_X=X$ та $K_Y=Y$	Відповідь буде позитивною лише при виконанні умов $K_X=X$ та $K_Y=Y$, інакше, якщо хоча б одна з рівностей не виконується, роздати подарунки буде неможливо.
$X+Y<N$	$Y\le K_Y$ та $X\le K_X$	Очевидно, в цьому випадку всі діти отримають свої подарунки.
	$Y<K_Y$ та $X>K_X$	За таких умов діти, які хотіли отримати непарну кількість цукерок, завжди будуть задоволені, але нас цікавлять усі діти. Тому необхідно ще визначити умову, за якої цукерки будуть видані "парним" дітям. Відомо, що при додаванні двох непарних чисел завжди отримаємо парне число. Отже, для того щоб "парні" діти були задоволені, слід перевірити, чи можна з залишку, який залишився після розподілу дарунків "непарним" дітям, скласти нові "парні" подарунки у необхідній кількості. Таким чином, якщо виконується умова $(K_Y-Y)\ div\ 2+K_X\ge X$, отримаємо позитивну відповідь, інакше – негативну.
	інакше	В будь-яких інших випадках задовольнити потреби всіх дітей неможливо.

Програмна реалізація наведеного алгоритму:

```
var a:array[1..10000] of integer;
    n,kx,ky,x,y,i:longint;
begin
   writeln('Count of presents: ');
   readln(n); kx:=0; ky:=0;
   writeln('Count sweets in the boxes: ');
   for i:=1 to n do begin
     readln(a[i]); if odd(a[i]) then inc(ky) else inc(kx);
   end;
   writeln('Input count of children: ');
   readln(x,y);
   if (x+y>n)
   then writeln('No')
   else if (x+y=n)
        then if  (kx=x) and (ky=y)
```


число M - кількість допустимих доз добрива ($1\le M\le 10000$), а в четвертому - M невід'ємних цілих чисел, що не перевищують 10000 і визначають дози добрив в грамах.

Вихідні дані. У єдиний рядок виведіть M чисел, кожне з яких дорівнює 1, якщо відповідна доза може бути отримана не більше ніж з двох пакетів, і 0, якщо її отримати не можна.

Зауваження. Узяті пакети повинні використовуватися повністю, не можна залишати в одному з них частину добрива.

Обмеження за часом: 1 сек. на тест

Приклад вхідних і вихідних даних.

Приклад введення:	*Приклад виведення:*
5 1 3 6 4 0 3 7 6 11	1 1 0

Аналіз. Створимо таблицю, в кожну K-ту клітинку якої будемо записувати 1, якщо K грам добрив можна отримати одним мішечком, 2 - двома, і 0 - якщо отримати не можливо. Слід звернути увагу на те, що максимальна вага мішечку з добривами 1000, тому двома мішечками не можливо отримати вагу більше 2000, отже таблиця має складатися з 2001 клітинки (1 необхідна для мішечка з вагою 0. Хоча це й не логічно, проте умова задачі дозволяє такий варіант). Зчитуємо з вхідного потоку N чисел (вагу пакетів у бабусі), і для кожного з них (k) виконуємо наступні дії: Якщо у k-ій клітинці ($mas[k]$) вже стоїть 1 (тобто пакет з такою вагою вже був), то в клітинку $mas[k+k]$ поставимо 2 (якщо в ній досі стоїть 0). Якщо ж в k-ій клітинці стоїть не 1, то перевіряємо всі можливі комбінації даного мішечка з тими, що вже були розглянуті, і якщо їх комбінація дає новий об'єм - помічаємо клітинку з новим об'ємом цифрою 2.

Після того, як всі мішечки бабусі було розглянуто, зчитуємо M чисел - необхідні дози добрив. Якщо зчитане число менше за 2001 та у відповідній клітинці стоїть 1 або 2, то виводимо 1, інакше - 0.

```
var mas:array[0..2000] of byte;
    k,i,j,n,m:longint;
begin
  readln(n);
  for i:=1 to n do
  begin
    read(k);
    if mas[k]=1 then
```

Приклад введення:	*Приклад виведення:*
100.00 2 20.00 45.00 3 10.00 20.00 35.00	55.00

Аналіз. Зрозуміло, що чим більша знижка у гривнях, тим менше коштів бабуся заплатить за мішок з добривом. Тому нам треба знайти найбільшу знижку у відсотках $max\{p_i\}$ та гривнях $max\{c_i\}$, після чого вибрати таку знижку, яка є найвигіднішою для бабусі, тобто максимальну з двох $max(max\{p_i\}, max\{c_i\})$.

Програмна реалізація наведеного алгоритму:

```
var i,m,n:longint;
    c,z,pp,cc:double;
begin
  readln(c); read(n);
  read(pp);
  for i:=2 to n do begin
    read(z);
    {визначення максімуму для знижок у відсотках}
    if pp<z then pp:=z;
  end;
  pp:=c*(1-pp/100); {знаходимо ціну у гривнях для
                     даного максимального відсотка}
  readln; read(m);
  read(cc);
  for i:=2 to m do begin
    read(z);
    {визначення максімуму  для знижок у гривнях}
    if cc<z then cc:=z;
  end;
  if (c-cc>pp) then write(pp:0:2) else write(c-cc:0:2);
end.
```

Задача 3. Добриво. Коли бабуся привезла додому куплений мішок з добривом, то відразу ж розфасувала його по N маленьким пакетам і склала їх акуратно в сараї. В журналі "Сам собі агроном" старенька вичитала які дози добрив можна застосовувати для яблунь. І тепер вона хоче для кожної з цих доз дізнатися, чи може вона отримати її, взявши не більше двох пакетів з сараю.

Завдання. Напишіть програму *fertil*, яка дасть відповідь на питання бабусі.

Вхідні дані. У першому рядку написано ціле число N ($1 \le N \le 100000$) - кількість пакетів. Другий рядок містить N невід'ємних цілих чисел, кожне з яких визначає вагу в грамах добрива у відповідному пакетику. Ці числа не перевищують 1000. У третьому рядку задається ціле


```
            then writeln('Yes')
            else writeln('No')
        else if (y<=ky) and (x<=kx)
            then writeln('Yes')
            else if (y<ky) and (x>kx)
                then if (ky-y) div 2+kx>=x
                    then writeln('Yes')
                    else writeln('No')
                else writeln('No');
end.
```

Задача 4. Астрономи жартують. В одного астронома було 3 однакових знімки зоряного неба. Один з його колег жартома домалював на одному зі знімків кілька зірок (більше за 0). Ще один колега стер кілька зірок на іншому знімку. Напишіть алгоритм, що допоможе бідному астрономові визначити, який із трьох знімків залишився без змін. Кожен знімок заданий квадратною таблицею розміру $N \times N$ (тобто складається з N рядків та N стовпців). Ненульові елементи позначають зірки, нульові - ділянки неба без зірок.

Аналіз. Так як кількість зірок на кожній з карт буде різною, то розв'язання задачі полягає у знаходженні кількості ненульових елементів кожної з таблиць, та пошуку серед цих чисел середнього.

```
type mas=array[1..100,1..100] of integer;

procedure vvod(letter:char; var m:mas);
var i,j:integer;
begin
  for i:=1 to n do
   for j:=1 to n do
    begin
     write(letter,'[',i,',',j,']=');
     readln(m[i,j])
    end;
end;

function stars_count(m:mas):integer;
var i,j,s:integer;
begin
 s:=0;
 for i:=1 to n do for j:=1 to n do
   if m[i,j]<>0 then s:=s+1;
 stars_count:=s;
end;

var a,b,c:mas;
    k_a,k_b,k_c,n:integer;
begin
```

```
26   writeln('Input size of table');  readln(n);
27   writeln('Input values from tables');
28   vvod('A',a);
29   vvod('B',b);
30   vvod('C',c);
31   k_a:=stars_count(a);
32   k_b:=stars_count(b);
33   k_c:=stars_count(c);
34   if (k_a<k_b) and (k_b<k_c) then writeln('True is B');
35   if (k_b<k_c) and (k_c<k_a) then writeln('True is C');
36   if (k_c<k_a) and (k_a<k_b) then writeln('True is A');
37 end.
```

Задача 5. «Із днем народження!». Четверо друзів Вінні-Пух, П'ятачок, Сова й Ослик, щороку відзначають свої дні народження (перший день народження в році у Вінні-Пуха, другий – у П'ятачка, третій – у Сови, і останній – в Ослика). У свій день народження іменинник запрошує інших друзів до себе на урочистий сніданок. Природно, що коли хтось іде на день народження до друга, то повинен принести подарунок. В якості подарунку уже давно в них прийнято дарувати один одному кульки. Кожний повинен принести другові хоча б одну кульку. Але для того, щоб іменинник був приємно здивований, і в нього був гарний настрій, він повинен одержати в сумі від усіх друзів неодмінно більшу кількість кульок, ніж було подаровано попередньому іменинникові в його день народження. Хоча й недобре передаровувати подарунки, але друзі цим часто користуються. Тому кожен може подарувати частину кульок (або навіть усі), які подарували у свій час йому. Якщо в когось не вистачає кульок, він повинен купувати їх у Кролика, який продає кульки за дуже високою спекулятивною ціною (купувати кульки один в одного, позичати або віддавати просто так, друзі не можуть). Зрозуміло, що друзі намагаються купувати (якщо це взагалі потрібно) якнайменше кульок. При цьому на самому початку в кожного з них є деякий запас кульок.

Напишіть алгоритм, що визначає найменшу кількість покупок, який необхідно зробити друзям, на N днів народження. Як приклад, розглянемо ситуацію, коли на самому початку кульок ні в кого немає й нам необхідно знайти кількість покупок для 2 днів народження. Для того, щоб прийти до Вінні-Пуху на день народження, П'ятачок, Сова й Ослик купують у Кролика по одній кульці (тобто всього 3). Отже, Вінні-Пух дуже задоволений, так як до цього ніхто не одержував цілих три кульки. На день народження П'ятачка, Вінні-Пух може подарувати 2 кульки з подарованих йому (і в нього залишиться ще 1), а Сова й Ослик повинні купити ще по одній кульці в Кролика (усього 2 покупки). Тоді П'ятачок


```
26     mem:=t div bs;
27   end;
28   while mem>0 do begin
29     x.c[mxlen-x.len]:=mem mod bs;
30     mem:=mem div bs;
31     inc(x.len);
32   end;
33
34   for i:=mxlen-x.len+1 to mxlen do
35   begin
36     str(x.c[i],z);
37     if (length(z)<bslen) and (i<>mxlen-x.len+1)
38       then
39         if x.c[i]=0
40           then write(zero[bslen])
41           else write(zero[bslen-length(z)]);
42     write(x.c[i]);
43   end;
44 end;
45
46 var n:int64; begin
47   read(n);
48   if n*(n+1) mod 3=0
49     then mult(n*(n+1) div 6,2*n+1)
50     else mult(n*(n+1) div 2,(2*n+1) div 3);
51 end.
```

Задача 2. Знижки. Продавши всі свої яблука, бабуся поспішила в магазин, щоб купити мішок добрива для яблунь. І буває ж щастя в житті! Саме цього дня в магазині проводилася акція, за якою на товар можна отримати одну із знижок за вибором. Знижки можуть бути як відносно вихідної (початкової) ціни у відсотках, так і абсолютні - у гривнях. Остаточна ціна округляється до копійок.

Завдання. Напишіть програму *discount*, що визначає мінімальну суму, за яку бабуся зможе придбати мішок добрив.

Вхідні дані. У першому рядку задається вихідна ціна C мішка добрив. Другий рядок починається з кількості N процентних знижок, за якою йдуть величини цих знижок p_i ($i = \overline{1, N}$). Третій рядок містить кількість M абсолютних знижок і величини цих знижок c_i ($i = \overline{1, M}$) у гривнях. Ціна і знижки задаються не більше ніж з двома знаками після крапки. Обмеження на величини - $0 \le C \le 1000000$, $0 \le M, N \le 10000$, $0 \le p_i \le 100$, $0 \le c_i \le C$.

Вихідні дані. Виведіть суму в гривнях, яку доведеться заплатити бабусі, з точністю до копійок (тобто з двома знаками після крапки). Обмеження за часом: 1 сек. на тест.

Приклад вхідних і вихідних даних.

Так як за умовою $1 \le N \le 100000000$, то в найгіршому випадку за формулою отримаємо:

$$\frac{100000000 \cdot 100000001 \cdot 200000001}{6} = 3\,333\,333\,383\,333\,333\,500\,000\,000.$$

Це число перевищує навіть число 9 223 372 036 854 775 807 (верхню межу типу *int64*), тому при розв'язанні даної задачі необхідно використовувати так звану "довгу арифметику". В нашому випадку треба знайти добуток довгих чисел (процедура *mult*).

Щоб звести кількість операцій до мінімуму, позбавимося від операції ділення, яка присутня у формулі, а саме ділення на 6. Зрозуміло, що в добутку двох чисел $n(n+1)$ одне з цих чисел обов'язково ділиться на 2, відповідно і сам добуток також ділиться на 2. Отже, залишилося визначити чи ділиться на 3 цей добуток. Якщо це так, то добуток $n(n+1)$ можна поділити без залишку на 6, в іншому випадку 3 є дільником числа $2n+1$. Розглянувши два варіанти окремо ми позбавляємося від операції ділення в загальній формулі $\frac{n(n+1)(2n+1)}{6}$ наступним чином:

```
48   if n*(n+1) mod 3=0
49     then mult(n*(n+1) div 6,2*n+1)
50     else mult(n*(n+1) div 2,(2*n+1) div 3);
```

Програмна реалізація наведеного алгоритму:

```
1  const mxlen=100;
2        bs=1000000; bslen=6;
3        zero:array [1..bslen] of string=
4    ('0','00','000','0000','00000','000000');
5  type longd=record
6                len:longint;
7                c:array [1..mxlen] of int64;
8              end;
9
10 procedure mult(a,b:int64); var i:longint;
11     t,mem:int64;
12     x:longd;
13     z:string;
14 begin
15   mem:=0; x.len:=0; i:=mxlen;
16   while a>0 do begin
17     x.c[i]:=a mod bs;
18     a:=a div bs; inc(x.len);
19     dec(i);
20   end;
21
22   for i:=mxlen downto mxlen-x.len+1 do
23   begin
24     t:=mem+x.c[i]*b;
25     x.c[i]:=t mod bs;
```


одержує 4 кульки й дуже задоволений тим, що він одержав більше ніж Вінні-Пух у свій день народження. Таким чином, кількість куплених кульок дорівнює 5, і меншої кількості покупок для 2 днів народження бути не може (якщо на початку ні в кого не було кульок).

Аналіз. Розглянемо дії друзів на 1-ий день народження. Нехай у Вінні-Пуха, П'ятачка, Сови та Ослика напередодні цього дня будуть $m[1]$, $m[2]$, $m[3]$ та $m[4]$ кульок відповідно. Вінні-Пуху необхідно подарувати мінімальну кількість кульок (щоб іншим не довелось більше дарувати), але не менш ніж по одній від кожного друга. Тобто Вінні на перший день народження за будь-яких умов отримає 3 кульки. Якщо в когось із друзів не вистачить кульок - доведеться необхідну кількість купувати у Кролика. Наступного дня народження (у П'ятачка) друзям необхідно подарувати на 1 кульку більше, тобто 4 штуки. Розглянемо дії друзів на k-ий день народження. Нехай у чотирьох друзів після минулих днів народження залишилось $m[1]$, $m[2]$, $m[3]$ та $m[4]$ кульок, причому перенумеруємо друзів таким чином, щоб кількість кульок k-го іменинника була в $m[1]$, $k+1$-ого - в $m[2]$ і т.д. та для зручності будемо вважати, що k-ий день народження - у Вінні-Пуха.

Оскільки першому імениннику було подаровано 3 кульки, другому 4 і т.д., то k-му імениннику необхідно подарувати $k+2$ кульки. З них три кульки подарують троє друзів іменинника - кожен по одній. Якщо в когось кульки не вистачає - доведеться купувати у Кролика. Якщо ще не всі необхідні кульки подаровано, починаємо збирати з друзів "хто скільки зможе". "Прожитковий мінімум" в кульках для друзів можна визначити просто: П'ятачок може віддати хоч усі свої кульки - наступний день народження буде у нього і він отримає достатню кількість кульок. Сові необхідно залишити 1 кульку - для подарунку П'ятачку, а потім у неї самої буде день народження. А Ослику потрібні 2 кульки - для подарунку П'ятачку та Сові. Все, що за межею "прожиткового мінімуму"друзі можуть спокійно дарувати. Збираємо необхідну кількість починаючи з П'ятачка та закінчуючи Осликом. Якщо ж і тепер подаровано ще не всі кульки - будемо обмежувати "прожитковий мінімум спочатку заберемо одну кульку у Ослика (якщо в нього вона є). Якщо є потреба - доведеться Ослику та Сові віддати останні кульки (якщо вони в них ще залишились). Якщо було подаровано ще не всі необхідні кульки, будь-хто з друзів посилається купувати у Ослика необхідну кількість.

Одна з можливих програмних реалізацій:

```
1 var m:array[1..5] of longint;
2     k,zz,t,j,i,n,count:longint;
```

```
begin
  count:=0;
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(n);
  readln(m[1],m[2],m[3],m[4]);
  close(input);
  for i:=1 to n do
  begin
    for j:=2 to 4 do if m[j]>0 then dec(m[j]) else inc(count);
    m[1]:=m[1]+(i+2); t:=i-1;
    for j:=2 to 4 do
    begin
      if (t>0) and (m[j]-(j-2)>0) then
      begin
        if t<=m[j]-(j-2) then
        begin
          m[j]:=m[j]-t;
          t:=0;
        end
        else
        begin
          t:=t-(m[j]-(j-2));
          m[j]:=j-2;
        end;
      end;
    end;
    if (t>0) and (m[4]>0) then begin dec(m[4]); dec(t); end;
    if (t>0) and (m[4]>0) then begin dec(m[4]); dec(t); end;
    if (t>0) and (m[3]>0) then begin dec(m[3]); dec(t); end;
    if t<0 then count:=count+t;
    m[5]:=m[1];
    for j:=1 to 4 do m[j]:=m[j+1];
  end;
  writeln(count);
  close(output);
end.
```

Если вы не можете решить задачу,
то всегда можете заглянуть в ответ.
Но пожалуйста, попытайтесь сначала решить ее самостоятельно;
тогда вы изучите больше и научитесь быстрее.
/Дональд Э. Кнут (Donald E. Knuth), Все про TeX (1983)/

10-11 клас

Задача 1. Підрахунок результату. Як відомо, на олімпіадах з інформатики перевірка завдань здійснюється автоматично за допомогою спеціальної програми, що компілює рішення учасника й запускає його на деякому наборі контрольних прикладів (тестів). Програма що, перевіряє,

чотирма сусідніми яблуками нижнього шару зверху було покладено ще по яблуку. Так само викладалися всі подальші шари аж до самого верхнього, який містив всього одне яблуко.

Завдання. Напишіть програму *apples*, яка визначає кількість яблук, що містить ця пірамідка.

Вхідні дані. У єдиному рядку записано одне число N ($1 \le N \le 100000000$) - сторона основи піраміди.

Вихідні дані. Виведіть одне ціле число - кількість яблук в піраміді. Обмеження за часом: 1 сек. на тест

Приклад вхідних і вихідних даних введення виведення.

Приклад введення:	*Приклад виведення:*
3	14

Аналіз. Очевидно, що для побудови першого шару піраміди з яблук потрібно рівно n^2 яблук, для другого шару - $(n-1)^2$, третього $(n-2)^2$ і так далі і лише одне яблуко потрібне для створення останнього верхнього шару. Тобто, для побудови всієї піраміди необхідно $n^2+(n-1)^2+\ldots+3^2+2^2+1$ яблук. Відомо, що ця сума $\sum\limits_{k=1}^{n} k^2 = 1+2^2+3^2+\ldots+(n-1)^2+n^2$ дорівнює $\frac{n(n+1)(2n+1)}{6}$, тобто розв'язок задачі зводиться до знаходження цього добутку.

Цю рівність досить легко отримати самостійно:

$$\sum_{k=1}^{n} k^3 + (n+1)^3 = \sum_{k=0}^{n}(k+1)^3 = \sum_{k=0}^{n}(k^3+3k^2+3k+1)^3 =$$

$$= \sum_{k=1}^{n} k^3 + 3\sum_{k=1}^{n} k^2 + 3\sum_{k=1}^{n} k + \sum_{k=1}^{n} 1 =$$

$$= \sum_{k=1}^{n} k^3 + 3\sum_{k=1}^{n} k^2 + 3\frac{(n+1)n}{2} + (n+1) \Leftrightarrow$$

$$\Leftrightarrow 3\sum_{k=1}^{n} k^2 = (n+1)^3 - 3\frac{(n+1)n}{2} - (n+1) =$$

$$= (n+1)\Big(n^2+2n+1-\frac{3}{2}n-1\Big) = n(n+1)\Big(n+\frac{1}{2}\Big) \Leftrightarrow$$

$$\Leftrightarrow \sum_{k=1}^{n} k^2 = \frac{1}{3}n(n+1)\Big(n+\frac{1}{2}\Big) = \frac{n(n+1)(2n+1)}{6}$$

блиця $A[1..N]$, в якій записано сили фігур, що стоять в ряду. Напишіть алгоритм, що визначає, скільки фігур на початку ряду потрібно перефарбувати, щоб різниця між сумарною силою чорних і білих фігур була мінімально можливою.

Аналіз. Нехай $a_1, a_2, \ldots, a_n$ - значення сил відповідних фігур. Тоді задача зводиться до знаходження такого числа k $(k < n)$, для якого виконується така умова $\left|\sum_{i=1}^{k} a_i - \sum_{i=k+1}^{n} a_i\right| \to min$. Так як, другу суму можна представити у вигляді $\sum_{i=k+1}^{n} a_i = \sum_{i=1}^{n} a_i - \sum_{i=1}^{k} a_i$. Підставивши, отримаємо: $\left|\sum_{i=1}^{k} a_i - \sum_{i=k+1}^{n} a_i\right| = \left|\sum_{i=1}^{k} a_i - \left(\sum_{i=1}^{n} a_i - \sum_{i=1}^{k} a_i\right)\right| = \left|2\sum_{i=1}^{k} a_i - \sum_{i=1}^{n} a_i\right| \to min$

Отже, при введенні значень сил фігур ми можемо одразу знаходити значення $\sum_{i=1}^{n} a_i$, а потім поступово будуємо суму $\sum_{i=1}^{k} a_i$.

Реалізація:

```
var f:array [1..100000] of byte;
    i,n,s,sk:longint;
begin
 s:=0;
 write('Input count figures: ');
 readln(n);
 writeln('Input forses of figures');
 for i:=1 to n do begin
   read(f[i]); s:=s+f[i];
 end;
 i:=1;  sk:=0;
 while sk+f[i]<s-(sk+f[i]) do begin
    sk:=sk+f[i]; inc(i);
 end;
 if abs(s-2*sk) > abs(s-2*(sk+f[i]))
   then writeln(i)
   else writeln(i-1);
end.
```

Оставьте книгу открытой на этой странице - это потрясет ваших родителей.

10-11 клас

Задача 1. Яблука. Бабуся продавала яблука на штуки. Щоб привернути увагу покупців, вона склала з них пірамідку: спочатку виклала квадрат розміру N (N рядів по N яблук у кожному), утворивши таким чином самий нижній шар. Наступний шар вийшов, коли між кожними

у результаті своєї роботи видає рядок, який містить по одному символу для кожного тесту:

"+" - рішення учасника видало правильну відповідь на відповідному тесті за час, що не перевищив припустимого (Yes);

"P" - результат вірний, але формат виводу відрізняється від зазначеного в умові й пам'ятці (Presentation Error);

"−" - рішення вклалося у відведений ліміт часу, але результат невірний (Wrong answer);

"R" - відбулася помилка на етапі виконання програми (Run-Time Error);

"T" - програма працювала довше відведеного ліміту часу (Time-Limit Exceeded);

"F" - відсутній у поточному каталозі файл результату (File Error);

"O" - формат вихідного файлу невірний, через що неможливо перевірити його правильність (Output Format Error);

"C" - програма учасника не компілюється (Compilation Error);

"N" - вихідний код програми учасника (на Pascal, C або C++) не знайдений у папці з його ім'ям (Not found program).

Інших символів програма, що перевіряє, видати не може. Очевидно, що якщо в рядку хоча б один раз зустрічається "C то всі символи в рядку будуть такими ж (якщо програма не відкомпілювалася при перевірці на одному тесті, то не відкомпілюється й при перевірці на будь-якому іншому). Аналогічне твердження справедливо й для символу "N". Інші ж символи можуть зустрічатися в цьому рядку в будь-якому сполученні. Кожен тест оцінюється деякою кількістю балів. Результат учасника визначається як сума балів за тести, на які його програма видає правильну відповідь ("+" або "P"). Але пам'ятайте, що за помилку подання "P" на тесті знімається 25% балів.

Завдання. Напишіть програму *calcres*, що визначає результат учасника по рядку, виданому програмою автоматичної перевірки.

Вхідні дані. Вхідний *calcres.dat* складається з трьох рядків. У першому рядку задається ціле число N ($0 \le N \le 1000$) - кількість тестів у наборі, на яких буде тестуватися рішення учасника. У другому рядку записані N цілих чисел, кожне з яких лежить у межах від 0 до 100 - кількість балів, яким оцінюються відповідні тести. І, нарешті, третій рядок - це рядок з N символів, що видає програма автоматичної перевірки.

Вихідні дані. У вихідний файл *calcres.sol* необхідно вивести результат учасника з точністю до двох знаків після десяткової крапки.

Приклад вхідних і вихідних даних

calcres.dat	*calcres.sol*
10 10 10 10 10 10 10 10 10 10 10 ++-+PPRTTT	45.00

Аналіз. Очевидно, що нам потрібно розглянути випадки в яких тест зараховано, хочаб і частково, тобто при значенні тесту "+" або "P" . В інших випадках жодного балу не буде зараховано. Але ще додатково можна опрацювати випадки, коли при перевірці були отримані символи "C" або "N" – в такому разі можна передчасно завершувати перевірку результатів, а відповідь буде рівна 0.

Програмна реалізація:

```
var b:array[1..1000] of integer;
    sum:real;
    test:char;
    i,n:integer;
begin
  sum:=0;
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(n);
  for i:=1 to n do read(b[i]);
  readln; i:=0;
  while not eoln do
    begin
      read(test); inc(i);
      case test of
      '+': sum:=sum+b[i];
      'P': sum:=sum+0.75*b[i];
      'C','N': begin sum:=0; break; end;
      end;
    end;
  close(input);
  if n=0 then writeln(0.1*n:0:2) else write(sum:0:2);
  close(output);
end.
```

Задача 2. Матрьошки. На полиці в магазині в рядок стоять матрьошки. В кожній з них поки нема інших, але існує можливість вставити одну матрьошку в іншу. Така можливість існує, якщо різниця висот складає рівно 1 см і іншої матрьошки такої ж висоти не знаходиться в середині.

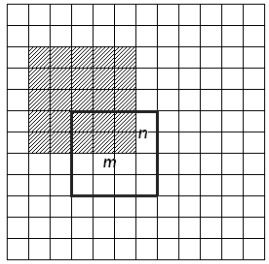
Завдання. Напишіть програму *nestdoll*, що допоможе покупцеві здійснити бажану покупку.

Вхідні дані. У першому рядку вхідного файлу *nestdoll.dat* записана

Задача 4. Різниця квадратів. На нескінченній шахівниці всі клітинки деякого квадрата розміру $a \times a$ були пофарбовані в червоний колір. Вам дозволили вирізати з цієї дошки квадрат будь-якого розміру. Напишіть алгоритм, що визначає чи можна зробити це так, щоб вирізаний квадрат містив рівно S червоних клітинок.

Зауваження. Сторони обох квадратів повинні проходити по лініях, що розділяють клітинки дошки.

Аналіз. Легко переконатися, що квадрат можна вирізати лише тоді, коли S можна представити у вигляді добутку $m \times n$, де $m, n \le a$, тобто коли в площині червоного квадрату можна побудувати прямокутник зі сторонами, розміри яких не перевищують a, в протилежному випадку квадрат вирізати не можна.



Отже задача зводиться до побудови всіх можливих пар (m, n), таких що $S = m \times n$. Під час програмної реалізації розв'язок можна оптимізувати, якщо шукати лише такі пари (m, n) в яких $m \le n$, всі інші пари будуть "симетричними" до знайдених $((m, n) = (n, m))$, та передчасно зупиняти пошук, якщо знайдеться хочаб одна пара, яка відповідає умові.

Реалізація:

```
var a,s,m,i,k:longint;
    r:string;
begin
  readln(a,s); r:='No';
  if a<trunc(sqrt(s))+1
    then k:=a else k:=trunc(sqrt(s))+1;
  for m:=1 to k do
    if (s mod m=0) and (s div m<=a)
    then begin r:='Yes'; break; end;
  writeln(r);
end.
```

Задача 5. Розділити навпіл.

Одного дня Василько та Петрик вирішили зіграти в шахи, але виявилось, що у них є лише N фігур і всі білого кольору. Тоді вони вишикували в ряд усі наявні фігури та вирішили перефарбувати в чорний колір перші декілька фігур, що стоять в ряду. Проте, щоб гра була цікавою, потрібно виконати перефарбування так, щоб сумарні сили білих і чорних фігур відрізнялися одна від одної на якомога меншу величину. Отже, вам задана цілочисельна та-

```
11   if abs(x1-x2)=abs(y1-y2)
12      then begin writeln('Bishop or Queen'); b:=true; end;
13   if (abs(x1-x2)*abs(y1-y2)=2)
14      then begin writeln('Knight'); b:=true; end;
15   if not b then writeln('No figure');
16 end.
```

Задача 3. Збери дошку. Відомо, що шахівницею називається дошка квадратної форми, кожна клітинка якої або біла, або чорна. Причому клітинки, що мають спільну сторону (сусідні по горизонталі або вертикалі), мають бути різного кольору. У вас є w білих клітинок і b чорних. Складіть алгоритм для визначення максимального розміру s дошки, яка може бути складена з цих клітинок.

Зауваження. Можна використовувати не всі клітинки.

Аналіз. Так як шахова дошка завжди квадратна, то потрібно визначити яка умова пов'язує кількість білих (w) та чорних (b) клітинок. Якщо розміри дошки $n \times n$, то при n – парному $w = b$, а n – непарному $|w - b| = 1$. Ці два випадки треба окремо розглянути в залежності від значень w та b.

Спочатку намагаємося побудувати дошку зі стороною n, де n-парне, тобто коли $w = b$. Легко бачити, що $n = \left[\sqrt{2 \cdot min(w, b)}\right]$, де позначення $[\,]$ - числова функція цілої частини числа (ант'є числа). Для непарного n: w та b повинні відповідати умові $|w - b| = 1$, а $n = \left[\sqrt{2 \cdot min(w, b) + 1}\right]$. Так як нам треба знайти дошку максимального розміру, яку можна скласти з w-білих та b-чорних клітинок, то з отриманих величин треба вибрати більшу:

$$N = max\left(\left[\sqrt{2 \cdot min(w, b)}\right], \left[\sqrt{2 \cdot min(w, b) + 1}\right]\right) =$$

$$= \left[\sqrt{2 \cdot min(w, b) + 1}\right].$$ Але треба розуміти, що для випадку $w = b$ ця формула не підходить, тому його потрібно обробляти окремо.

Програмна реалізація:

```
1  var w,b,mn:longint;
2  begin
3    readln(w,b);
4    if w=b
5      then writeln(trunc(sqrt(2*w)))
6      else begin
7        if w>b then mn:=b else mn:=w;
8        writeln(trunc(sqrt(2*mn+1)));
9      end;
10 end.
```


кількість матрьошок N ($1 \le N \le 10000$). У другому рядку записані N цілих чисел, що визначають висоти відповідних матрьошок у сантиметрах. Кожне число перебуває в межах від 1 до 10^9.

Вихідні дані. У перший рядок вихідного файлу *nestdoll.sol* необхідно вивести максимальну кількість матрьошок, які можуть бути послідовно вставлені одна в одну, а другий рядок повинен містити номера таких матрьошок (у довільному порядку). Обмеження за часом: 0,5 секунди на тест.

Приклад вхідних і вихідних даних

nestdoll.dat	*nestdoll.sol*
10 20 3 5 3 1 7 4 2 8 10	5 2 3 5 7 8

Аналіз. Нехай вхідні дані складаються з двох рядків. Перший – це кількість матрьошок в магазині, а другий – висоти відповідних матрьошок. Тоді задача зводиться до знаходження найдовшої монотонної послідовності висот матрьошок з різницею висот рівно в 1 см. Пошук такої послідовності зручно буде зробити в відсортованому масиві. Так як в результаті роботи програми необхідно вивести номери матрьошок, які необхідно купити, то вхідні дані будемо зберігати в двовимірному масиві, де перша компонента – це висота матрьошки, а друга - її початковий порядковий номер на полиці.

Метод пошуку найбільшої монотонної послідовності полягає в перевірці умови монотонності з підрахунком початку послідовності та її довжини. Найдовша поточна послідовність порівнюється зі знайденою, таким чином, наприкінці роботи отримаємо найдовшу монотонну послідовність.

Реалізація наведеного алгоритму може бути наступною:

```
1  var mass: array [1..10000,1..2] of longint;
2      n,i,j,mdl,tdl,n_tdl,n_mdl:integer;
3      r:longint; pr:boolean;
4  begin
5   assign(input,'nestdoll.dat'); assign(output,'nestdoll.sol');
6   reset(input); rewrite(output);
7   Readln(n);
8    for i:=1 to n do begin
9      read(mass[i,1]); mass[i,2]:=i;
10   end;
11   for i:=1 to n-1 do
12    for j:=i+1 to n do
13    if mass[i,1]=mass[j,1]
14     then begin mass[i,1]:=-1; n:=n-1; end;
15   pr:=true;
16   while pr do
```

```
    begin
      pr:=false;
      for i:=1 to n-1 do
       if mass[i,1]>mass[i+1,1] then
        begin
          r:=mass[i,1]; mass[i,1]:=mass[i+1,1];
          mass[i+1,1]:=r; r:=mass[i,2];
          mass[i,2]:=mass[i+1,2]; mass[i+1,2]:=r;
          pr:=true;
        end;
    end;
  n_tdl:=1;tdl:=1; n_mdl:=0;mdl:=0;
  for i:=1 to n do
   begin
     if mass[i,1]+1<>mass[i+1,1]
      then
       begin
        if tdl>=mdl
         then begin n_mdl:=n_tdl; mdl:=tdl; end;
         tdl:=1; n_tdl:=i+1;
       end
      else tdl:=tdl+1;
   end;
  writeln(mdl);
  for i:=n_mdl to n_mdl+mdl-1 do
    write(mass[i,2],' ');
    writeln;
 close(input);    close(output);
end.
```

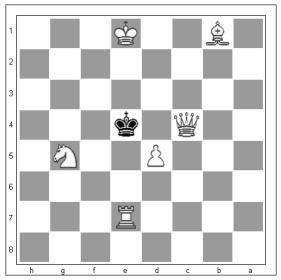
Задача 3. Сепарабельна матриця. Матриця $A(M \times N)$ називається сепарабельною, якщо вона визначається як сума двох векторів розмірностей M та N відповідно. Тобто $A[i,j] = B[i] + C[j]$.

Завдання. Напишіть програму *separate*, що визначає чи є задана матриця сепарабельною.

Вхідні дані. У першому рядку вхідного файлу *separate.dat* записані два цілих числа M і N ($1 \le M, N \le 1000$). У наступних M рядках записано по N цілих чисел, що визначають відповідні елементи матриці A. Кожне число не перевершує за модулем 10^8.

Вихідні дані. У перший рядок вихідного файлу *separate.sol* необхідно вивести слово Yes, якщо матриця A сепарабельна, і No у протилежному випадку. Якщо матриця A сепарабельна, то в другому рядку виведіть поелементно масив B, а в третьому – масив C, на які розбивається матриця A. Якщо існує кілька варіантів вибору масивів B і C, виберіть із них такий, щоб всі елементи були цілими числами, та не перевищували $2 \cdot 10^9$ за абсолютною величиною.

Король не може загрожувати іншому королю – позиція у грі, при якій два королі опиняються в сусідніх клітках, недопустима в шахах.



Аналіз. Нехай сторони дошки занумеровані від 1 до 8 по вертикалі та горизонталі відповідно, починаючи з лівого верхнього кута. Також домовимось, що бік чорних знаходиться у верхній частині дошки, тобто білі будуть починати свій рух знизу. Тоді, для того щоб пішак мав змогу напасти на чорного короля, він повинен знаходитись на одну горизонталь нижче та зліва або зправа відносно короля на одну клітину. Тобто, для пішака можна побудувати таку умову, за якої він матиме можливість об'явити королю шах – це $x_1 - x_2 = 1$ та $|y_1 - y_2| = 1$. Якщо пішак може атакувати з такої відстані, то це може зробити і ферзь. Легко бачити, що для слона відповідна умова можливості нанесення удару чорному королю є $|x_1 - x_2| = |y_1 - y_2|$. Оскільки по діагоналі може рухатись і ферзь, то ця умова відповідає і йому. Тура може напасти на короля лише у випадку, коли вони стоять на одній горизонталі або вертикалі: $x_1 = x_2$ або $y_1 = y_2$, ця ж умова також підходить і до можливостей ферзя. Умови для коня (як відомо, найхитрішої фігури) представимо у вигляді сукупності:

$$\left[\begin{array}{l} \begin{cases} |x_1 - x_2| = 2 \\ |y_1 - y_2| = 1 \end{cases} \\ \begin{cases} |x_1 - x_2| = 1 \\ |y_1 - y_2| = 2 \end{cases} \end{array}\right.$$

Дану сукупність можна представити у вигляді лише однієї умови, яка акумулює всі властивості сукупності, $|x_1 - x_2| \cdot |y_1 - y_2| = 2$. Розглянувши ці всі випадки, отримаємо відповідь на задачу (*в програмі використані англійські назви фігур: King (Король), Queen (Ферзь), Rook (Тура), Bishop (Слон), Knight (Кінь), Pawn (Пішак)*):

```
var x1,x2,y1,y2:integer;
    b:boolean;
begin
  b:=false;
  write('Position of the King: ');readln(x1,y1);
  write('Position of Figures: ');readln(x2,y2);
  if (x1-x2=1) and (abs(y1-y2)=1)
     then begin  writeln('Pawn or Queen'); b:=true; end;
  if (x1=x2) or (y1=y2)
     then begin writeln('Rook or Queen'); b:=true; end;
```

II спосіб. Можливо, цей спосіб буде більш прозорим. Відмінністю від першого способу буде лише визначення кількості ударів при $M > K$. Легко бачити, що можна побудувати звичайний циклічний алгоритм, в якому за кожну ітерацію враховувати кількість зрубаних та регенерованих голів $N = N - M + K$, до тих пір, поки це можливо, тобто поки $N > M$.

Зауваження. Треба розуміти, що другий спосіб при дуже великому значенні N та малих значеннях M і K ($M > K$) буде виконуватися значно довше, оскільки ми поступово зменшуємо кількість голів та підраховуємо удари, а не одразу шукаємо відповідне значення використовуючи операцію ділення.

Реалізація наведених алгоритмів на мові Pascal:

{I спосіб}

```
var m,n,k,c:longint;
begin
  readln(n,m,k); c:=0;
  if m>=n
  then c:=1
  else if m>k
       then begin
         n:=n-m;
         c:=n div (m-k)+1;
         c:=c+ord(n mod (m-k)>0);
       end
       else c:=-1;
  writeln(c);
end.
```

{II спосіб}

```
var m,n,k,c:longint;
begin
  readln(n,m,k); c:=0;
  if m>=n
  then c:=1
  else if m>k
       then begin
         while n>m do begin
           n:=n-m+k;
           c:=c+1;
         end;
         if n>0 then c:=c+1;
       end
       else c:=-1;
  writeln(c);
end.
```

Задача 2. Оголошення шаху.

На стандартній шахівниці розміру 8×8 у клітинці з координатами (x_1, y_1) стоїть чорний король.

Складіть алгоритм знаходження якої-небудь білої фігури, яку можна поставити на клітинку з заданими координатами (x_2, y_2) і яка загрожувала б при цьому королю.

Зауваження. Нагадаємо, що пішак загрожує клітинкам, розташованим по діагоналі від нього на наступній горизонталі, тура загрожує всім клітинам, розташованим на одній вертикалі або горизонталі з нею, кінь – клітинам, зміщеним відносно нього на 2 клітини вздовж однієї з осей і на 1 вздовж іншої, слон – клітинам, розташованим на одній з ним діагоналі, а ферзь – клітинам на одній з ним вертикалі, горизонталі або діагоналі.

Обмеження за часом: 1 секунда на тест.

Приклад вхідних і вихідних даних

separate.dat	*separate.sol*
2 2 1 1 2 3	No
2 3 3 2 1 4 3 2	Yes 1 2 2 1 0

Аналіз. Зрозуміло, що якщо $M = 1$ або $N = 1$, то відповідь буде завжди позитивною. Розглянемо випадок $M = 2$, $N = 2$. Для цього розв'яжемо систему лінійних рівнянь $B + C = A$ відносно невідомих B та C. Тобто

$$\begin{pmatrix} b_1 + c_1 & b_1 + c_2 \\ b_2 + c_1 & b_2 + c_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}.$$

Розв'язуючи методом Гауса, отримуємо умову, за якої система взагалі може мати розв'язки, а саме $a_{11} - a_{21} = a_{12} - a_{22}$. За цієї умови система має загальний розв'язок:

$$b_1 = a_{11} - a_{21} + a_{22} - c_2$$

$$b_2 = a_{22} - c_2$$

$$c_1 = a_{21} - a_{22} + c_2$$

$$c_2 = c_2$$

Знайшовши елементи b_1 та c_1, послідовно знаходимо решту елементів векторів B та C. Наступним кроком програми буде перевірка сепарабельності матриці A.

Приклад одного з варіантів реалізації наступний:

```
program separabel;
const mc=1000; nc=1000;
var a:array[1..mc,1..nc]of integer;
    b:array[1..mc] of integer;
    c:array[1..nc] of integer;
    m,n,i,j,t:integer;
    pr:boolean;
begin { Основна програма }
  assign(input,'separate.dat');
  assign(output,'separate.sol');
  reset(input); rewrite(output);
  readln(m,n);
```

```
13   for i:=1 to m do begin
14     for j:=1 to n do read(a[i,j]);
15     readln;
16   end;
17   if a[1,1]-a[2,1]=a[1,2]-a[2,2] then
18    begin {Перевірка необхідної умови}
19     if (m>1) and (n>1) then
20      begin
21       c[2]:=0;    b[1]:=a[1,1]-a[2,1]+a[2,2]-c[2];
22       b[2]:=a[2,2]-c[2];   c[1]:=a[2,1]-a[2,2]+c[2];   t:=3;
23      end
24      else  begin  b[1]:=1; c[1]:=a[1,1]-1;   t:=2;  end;
25     for i:=t to m do  b[i]:=a[i,1]-c[1];
26     for i:=t to n do c[i]:=a[1,3]-b[1];
27     pr:=true;
28     for i:=1 to m do
29      for j:=1 to n do
30       if a[i,j]<>b[i]+c[j] then pr:=false;
31     if pr then  begin
32       writeln('Yes');
33       for i:=1 to m do write(b[i],' '); writeln;
34       for i:=1 to n do write(c[i],' '); end
35      else
36      writeln('No');
37    end
38   else
39    writeln('No');
40   close(input); close(output);
41 end.
```


2008 рік

Прежде чем решить задачу, подумай, что делать с ее решением.
/Д. Пойа (G. Polya), Как решить задачу (1961)/

8-9 клас

Задача 1. Бій з драконом. У шаховому королівстві завівся N-головий шаховий дракон, який нападав на інші фігури та з'їдав їх. Тоді шаховий король дав шаховому ферзю шаховий щит і шаховий меч і наказав знищити таку дивну агресивну фігуру. Осідлав шаховий ферзь свого вірного шахового коня та поскакав до драконова лігвища. І ось зійшлися ферзь і дракон у шаховому бою, який відбувається таким чином. Спочатку атакує ферзь – він може відрубати одним ударом свого шахового меча будь-які голови дракона, але не більш ніж M за раз. Потім атакує дракон своїм вогненним подихом, ферзь ховається за шаховим щитом, а за цей час у дракона виростають K нових голів. Після цього знову атакує ферзь і так далі. Бій продовжується до тих пір, поки у дракона не залишиться жодної голови. В цьому випадку дракон гине, а ферзь повертається до свого короля з перемогою. Але бій може й не закінчитися ніколи. Складіть алгоритм, що визначає, чи зможе ферзь здолати дракона, і якщо так, то яке мінімальне число ударів йому буде потрібно для цього.

Аналіз.

І спосіб. Надамо алгоритм розв'язку задачі у вигляді таблиці, в якій розглянуто всі можливі випадки:

Якщо	*Відповідні дії*	
$M \geq N$	Зрозуміло, що при такій умові, мінімальна кількість ударів яку нанесе ферзь дорівнює $C = 1$	
інакше ($M < N$)	$M > K$	При будь-якій початковій кількості голів, при кожному ударі, кількість голів дракона буде зменшуватися. Одразу врахуємо останній удар, який можна зробити на повну силу, та при якому регенерації голів вже не відбудеться, тобто ферзю потрібно буде на попередніх ударах зрубати $N = N - M$ голів. Мінімальна кількість ударів для цього $C = N\ div\ (M - K)$, але треба пам'ятати, якщо $N\ mod\ (M - K) > 0$ тоді ферзю доведеться зробити ще один "контрольний" удар, після якого дракону вже не вижити.
	інакше ($M \leq K$)	Очевидно, що ферзь не зможе подолати дракона з такою шаленою швидкістю регенерації голів. В такому випадку відповідь охарактеризуємо числом $C = -1$.