ОЛІМПІАДА З ІНФОРМАТИКИ У МІСТІ КИЄВІ У 2013–2014 НАВЧАЛЬНОМУ РОЦІ

Кордубан Дмитро Олександрович,
аспірант Інституту програмних систем Національної академії наук України.
Мисак Данило Петрович,
керівник гуртка СШ №52 м. Києва.
Рудик Олександр Борисович,
доцент Київського університету імені Бориса Грінченка.

Стаття містить умови завдань II (районного) і завдань III (міського) етапу олімпіади з основ інформатики й обчислювальної техніки у місті Києві у 2013–2014 навчальному році й авторські розв'язання цих завдань. Публікацію адресовано учням класів з поглибленим вивченням математики, учасникам олімпіад з інформатики, студентам математичних спеціальностей, учителям і викладачам вищих навчальних закладів.

Цього навчального року орієнтовні завдання II етапу (упорядник — Данило Мисак) не містили завдань відбірково-тренувальних зборів і були краще узгоджені з рівнем підготовки переважної кількості учасників олімпіади. Розглянемо їх детально.

I. УМОВИ ЗАВДАНЬ II ЕТАПУ

Максимальна оцінка за кожну з чотирьох задач — 100 балів. Для всіх задач обмеження на час — 1 секунда/тест; обмеження на пам'ять — 256 МБ.

1. Письмо

Назва програми: writing.pas / writing.cpp

На уроці письма й каліграфії Петрик один або кілька разів поспіль виписав у зошиті одне й те саме натуральне число. У результаті утворилося шестицифрове число n. Допоможіть учительці з'ясувати, яке найменше число міг виписувати Петрик.

Вхідні дані. У вхідному файлі вказане шестицифрове натуральне число n.

Вихідні дані. У вихідний файл виведіть найменше натуральне число, яке міг виписувати Петрик.

Приклади

Вхідний файл writing.in	Вихідний файл writing.out
333333	3
525252	52
171171	171
240982	240982

2. Література

Назва програми: reading.pas / reading.cpp

Щоб уникнути чергової двійки з літератури, Петрик має виконати домашнє завдання: вибрати з хрестоматії два різних твори загальним обсягом s сторінок і прочитати їх. Знаючи обсяг кожного з n творів, що є в хрестоматії, допоможіть хлопцю з'ясувати, скільки він має варіантів вибору.

Вхідні дані. У першому рядку вхідного файлу вказано натуральні числа s та n, не менші за 2. У другому рядку записано n натуральних чисел — обсяги (кількості сторінок) відповідних творів із хрестоматії. Усі числа у вхідному файлі (включно з числами s та n) не перевищують 200 000.

Вихідні дані. У вихідний файл виведіть єдине число — кількість способів вибрати з хрестоматії пару творів загальним обсягом рівно s сторінок. Відомо, що ця кількість не перевищує 10^9.

Приклади

Вхідний файл reading.in	Вихідний файл reading.out
4 5 2 2 3 2 1	4
10 3 6 2 10	0

Пояснення до прикладів

У першому прикладі Петрик може вибрати один із чотирьох варіантів: прочитати перший і другий твори з хрестоматії; або перший і четвертий; або другий і четвертий; або третій і п'ятий.

У другому прикладі жодні два твори не дають у сумі рівно 10 сторінок.

3. Фізкультура

Назва програми: pe.pas / pe.cpp

На уроці фізкультури учні вишукувалися у шеренгу в деякому зручному для них порядку. Конче потрібно, однак, щоб усі стояли за зростом — у порядку від найвищого учня до найнижчого. Для цього вчитель може один або кілька разів зробити таке: назвати ім'я деякого учня і попросити його перейти в кінець шеренги. За яку найменшу кількість подібних дій учителю вдасться впорядкувати шеренгу бажаним чином?

Зріст кожного з n учнів відомий. Якщо кілька з них однакового зросту, то між собою такі учні в кінцевому розташуванні можуть стояти в довільному порядку (на вибір учителя). За бажання вчитель може переставляти одного й того самого учня кілька разів.

Вхідні дані. У першому рядку вхідного файлу вказано натуральне число n, не менше за 2 і не більше за 200 000. У другому рядку записано n натуральних чисел, що не перевищують $2 \cdot 10^9$, — зріст кожного з n учнів у порядку, в якому вони стоять у шерензі на початку уроку.

Вихідні дані. У вихідний файл виведіть єдине число — найменшу кількість перестановлянь, які доведеться зробити вчителю, щоб упорядкувати шеренгу за незбільшенням зросту учнів.

Приклади

Вхідний файл pe.in	Вихідний файл pe.out
7 126 145 141 134 130 141 134	3
3 173 172 169	0

Пояснення до прикладів

У першому прикладі вчитель може діяти так: спочатку переставити в кінець шеренги першого з двох учнів зросту 134 далі переставити учня зросту 130, а потім — 126. Швидше ніж за три дії впорядкувати шеренгу вчителю не вдасться.

У другому прикладі учні відразу стоять у потрібному порядку. Переставляти нікого не потрібно.

4. Математика

Назва програми: math.pas / math.cpp

Щоб заробити чергові дванадцять балів з математики, Петрик має розв'язати багато однотипних задач на переливання. Типова задача з Петрикового підручника має такий вигляд.

Є три порожні посудини місткістю a, b та c літрів. За одну операцію можна виконати одну з трьох дій.

- Повністю заповнити одну з посудин водою з джерела. До операції посудина може бути або порожньою, або вже частково наповненою. Кількість води у джерелі необмежена.
- Вилити з однієї з посудин усю воду, що в ній міститься. До операції посудина не обов'язково має бути повністю заповненою.
- Перелити з однієї посудини в іншу рівно стільки води, щоб або посудина, куди переливають воду, повністю заповнилася, або — якщо води для цього недостатньо — посудина, звідки переливають воду, спорожніла.

Операції кожного з трьох типів можна виконувати як завгодно багато разів. Треба з'ясувати, за яку найменшу кількість операцій і чи можливо взагалі «відміряти» n літрів, тобто зробити так, щоб принаймні в одній посудині опинилося рівно n літрів води.

Вхідні дані. У першому рядку вхідного файлу записано три натуральних числа a, b та c, а у другому рядку — натуральне число n. Усі чотири числа не перевищують 100.

Вихідні дані. У вихідний файл виведіть єдине число — найменшу кількість операцій, які потрібно провести з посудинами, щоб в одній з них опинилося рівно n літрів води. Якщо цього досягти неможливо, виведіть число 0.

Приклади

Вхідний файл math.in	Вихідний файл math.out
4 6 9 7	4
3 10 3 1	5
5 4 7 9	0

Пояснення до прикладів

У першому прикладі відміряти 7 літрів можна щонайменше за 4 кроки. Наприклад, так:

1. Налити воду в 4-літрову посудину.
2. Перелити з неї усі 4 літри води в 6-літрову посудину.
3. Налити воду в 9-літрову посудину.
4. Перелити з неї 2 літри в 6-літрову посудину (та заповниться). У 9-літровій посудині залишиться рівно 7 літрів води.

У другому прикладі відміряти 1 літр можна щонайменше за 5 операцій. Приміром, так.

1. Налити воду в 10-літрову посудину.
2. Перелити з неї 3 літри в першу посудину.
3. Перелити з неї ж іще 3 літри в третю посудину. У 10-літровій посудині залишиться 4 літри води.
4. Вилити воду з першої 3-літрової посудини.
5. Перелити 3 літри води з 10-літрової посудини в порожню 3-літрову. У 10-літровій залишиться 1 літр.

У третьому прикладі кількість літрів, яку потрібно відміряти, перевищує місткість кожної з трьох посудин. Тож відміряти відповідну кількість води, очевидно, не вдасться.

II. ІДЕЇ РОЗВ'ЯЗАННЯ ЗАВДАНЬ II ЕТАПУ

1. Письмо

З умови задачі випливає, що шукане число може бути лише одноцифровим, двоцифровим, трицифровим або шестицифровим.

Прямий спосіб розв'язання задачі передбачає розбиття зчитаного числа n на цифри (або посимвольне його зчитування) і порівняння між собою пар цифр:

- якщо всі цифри числа n однакові, то відповідь одноцифрова і дорівнює першій цифрі числа n;
- інакше, якщо третя і п'ята цифри числа n дорівнюють першій, а четверта і шоста — другій, то відповідь двоцифрова і дорівнює числу, яке утворюють перші дві цифри числа n;
- інакше, якщо четверта цифра числа n дорівнює першій, п'ята — другій, а шоста — третій, то відповідь трицифрова і дорівнює числу, яке утворюють перші три цифри числа n;
- інакше відповідь шестицифрова і дорівнює самому числу n.

Математичний спосіб розв'язання задачі (з використанням подільності) ґрунтується на спостереженні, що шестицифрове число n дорівнює добутку числа-відповіді й одного з таких чотирьох чисел: 111 111 (якщо відповідь одноцифрова), 10 101 (якщо відповідь двоцифрова), 1001 (якщо відповідь трицифрова) або просто 1 (якщо відповідь шестицифрова). Отже, відповіддю є частка числа n і першого з чисел 111 111, 10 101, 1001, 1, на яке n ділиться без остачі.

2. Література

Безпосередній перебір принесе лише частковий бал, бо для великих n програма не встигне за відведений час (секунду) перебрати всі можливі пари чисел. Замість цього можна скористатися іншим підходом. Для кожного числа від 1 до 200 000 слід підрахувати, скільки творів мають даний обсяг: завести масив cnt, у комірці $cnt[p]$ якого буде зберігатися кількість творів обсягу p ($1 \leqslant p \leqslant 200\,000$), і після зчитування кожного наступного числа збільшувати відповідну комірку масиву на 1. Далі через $[(s-1)/2]$ ми позначаємо найбільше ціле число, що не перевищує $(s-1)/2$. Для всіх значень p у межах від 1 до $[(s-1)/2]$ включно до-

буток $cnt[p]\cdot cnt[s-p]$ дорівнюватиме кількості пар творів, один із яких має обсяг p, а інший — обсяг $s-p$ сторінок; при цьому $p<s-p$. Сума всіх $[(s-1)/2]$ таких добутків буде кількістю варіантів вибрати пару творів сумарним обсягом s сторінок за умови, що обсяги цих двох творів різні. Залишається врахувати пари творів однакового обсягу: якщо s непарне, таких пар немає, а інакше їх рівно

$$\frac{cnt[s/2]\cdot(cnt[s/2]-1)}{2}.$$

3. Фізкультура

Передусім зауважимо, що переставляти одного й того самого учня кілька разів учителю не знадобиться: з послідовності дій можна вилучити всі переставляння даного учня, крім останнього, не збільшивши загальної кількості дій і не змінивши кінцевого результату. Стане в пригоді й таке спостереження: усі учні, яких учитель не переставляв у кінець шеренги, у підсумку будуть стояти на її початку, причому в тому ж відносному порядку, у якому вони стояли до переставлянь. Отже, це найвищі з усіх учнів, що стоять у порядку незбільшення зросту. І навпаки: якщо в початковій шерензі ми зафіксуємо і не будемо рухати кількох найвищих учнів, що стоять у порядку незбільшення зросту, а інших переставлятимемо в кінець шеренги в порядку від найвищого до найнижчого, то в підсумку впорядкуємо учнів як вимагається. Найменшій кількості переставлянь відповідає при цьому найдовша послідовність найвищих учнів, що стоять у порядку незбільшення зросту: якщо таких учнів k, то шукана кількість дій дорівнює $n-k$. Залишається знайти найдовшу таку послідовність, тобто число k.

Розв'язання сортуванням. Послідовність чисел, подану у вхідному файлі, можна занести відразу у два масиви, причому після зчитування один із них слід відсортувати за незростанням. Рухаючись зліва направо у невідсортованому масиві, треба додавати до лічильника (змінної k) одиницю кожного разу, коли натрапляємо на чергове число з відсортованого масиву.

Відзначимо, що просте сортування за квадратичний час дасть лише частковий бал, тож потрібно використати впорядкування злиттям або інший швидкий метод сортування масиву.

Розв'язання зі стеком. Назвімо найдовшу послідовність найвищих учнів, що стоять у порядку незбільшення зросту, *суперпідпослідовністю* даної послідовності. Суперпідпослідовність можна побудувати за один прохід вхідного масиву, тобто відразу під час зчитування файлу. Зчитавши перші p чисел вхідної послідовності, ми матимемо такі дані:

- стек s, що містить суперпідпослідовність послідовності з уже зчитаних p чисел (якщо $p=0$, стек порожній);
- змінну m, яка дорівнює найбільшому з тих уже зчитаних чисел, які не входять у стек s (якщо $p=0$, покладемо $m=0$).

Очевидно, за таких умов після зчитування всіх n чисел змінна s буде містити саме ту послідовність, яку й потрібно відшукати. Залишається розібратися, як підтримувати змінні s та m в актуальному стані. Після зчитування $(p+1)$-го числа, яке позначимо через h, їх слід оновити так:

1) зі стека s видалити всі числа, що строго менші за h. При цьому, якщо доведеться, відповідно збільшити значення m;

2) якщо h не менше за m, додати його в стек; інакше проігнорувати.

Покажемо, що якщо після p-го кроку стек s містить суперпідпослідовність перших p чисел, то після $(p+1)$-го кроку стек міститиме суперпідпослідовність нової послідовності з $(p+1)$ числа. Насамперед, як випливає зі схеми роботи алгоритму, жодне число зі стека не може бути меншим за поточне значення змінної m. Тепер випишемо можливі варіанти:

- число h є строго більшим за останній (найменший) елемент стека;
- число h не перевищує останній елемент стека, але й не менше за m;
- число h є меншим за m.

Можна переконатися, що в кожному з цих трьох випадків алгоритм справді, як потрібно, оновлює вміст стека.

Насамкінець додамо, що алгоритм працює лінійний час: кожне число послідовності додають у стек і видаляють з нього щонайбільше по одному разу. Саме в цьому полягає перевага цього способу розв'язання перед концептуально простішим методом сортування.

4. Математика

Станом назвімо впорядковану трійку чисел $(x; y; z)$, де x, y та z — кількості літрів води у першій, другій і третій посудинах відповідно. Якщо зі стану А за одну операцію з посудинами можна дістати стан Б, казатимемо, що стан Б *досяжний* зі стану А.

Для розв'язання задачі можна використати поданий нижче алгоритм, аналогічний пошуку в ширину на графі.

1. Заведемо чергу, у якій зберігатимемо стани — трійки цілих чисел. На практиці можна використати три окремих черги, елементами яких є цілі числа (відповідні компоненти трійок), і використовувати ці черги паралельно. Додамо в чергу початковий стан (0; 0; 0).

2. Розглянемо стан А, що є на даний момент першим у черзі, і видалимо його з черги. Додамо в чергу всі стани, які досяжні зі стану А і які ми не додавали в чергу раніше. Запам'ятаємо, що кількість операцій, необхідна для одержання з початкового стану (0; 0; 0) кожного з цих станів, на 1 більша, ніж для самого стану А. Щоби запам'ятати цю кількість, можна використати, наприклад, тривимірний масив, індексами якого слугує трійка чисел, що задає стан, а значенням комірки є відповідна кількість операцій. Цей же масив дозволить визначати, чи ми вже додавали в чергу той чи інший стан.

3. Якщо до черги було додано трійку, хоча б один компонент якої дорівнює n, відповідь знайдено (це кількість операцій, необхідна для досягнення даної трійки). Якщо черга спорожніла, а відповідь так і не знайдено, то відміряти n літрів води неможливо. Якщо ж черга ще не спорожніла і відповідь поки не знайдено, повертаємось до другого кроку алгоритму.

ІІІ. АВТОРСЬКІ РОЗВ'ЯЗАННЯ ЗАВДАНЬ ІІ ЕТАПУ

1. Письмо

```
{ Free Pascal }
Var
  f: text;
  n, ans, i: longint;
  d: array[1..4] of longint =
  (111111, 10101, 1001, 1);
begin
  assign(f, 'writing.in');
  reset(f);
  read(f, n);
  close(f);

  for i := 1 to 4 do
    if n mod d[i] = 0 then begin
      ans := n div d[i];
      break;
    end;

  assign(f, 'writing.out');
  rewrite(f);
  writeln(f, ans);
  close(f);
end.
```

2. Література

```
{ Free Pascal }

const
  M = 200000; // Максимальна можлива
        // кількість сторінок у творі
var
  f: text;
  s, n, ans, i, p: longint;
  cnt: array[1..M] of longint;
 // cnt[p] буде дорівнювати кількості
 // творів обсягу p
begin
 // Ініціалізація масиву:
  for p := 1 to M do
    cnt[p] := 0;

  assign(f, 'reading.in');
  reset(f);
  read(f, s, n);
  for i := 1 to n do begin
    read(f, p);
    inc(cnt[p]); // Якщо черговий твір
          // має обсяг p сторінок,
         // збільшуємо cnt[p] на один
  end;
  close(f);

  ans := 0;

 // Збільшуємо відповідь на кількість пар
 // творів, один із яких має обсяг
 // p сторінок, а інший — s-p сторінок,
 // де p пробігає всі значення
 // від 1 до [(s-1)/2]:
  for p := (s - 1) div 2 downto 1 do
    inc(ans, cnt[p] * cnt[s - p]);

 // Якщо s парне, збільшуємо відповідь
 // на кількість пар творів, кожен із яких
 // містить рівно по s/2 сторінок:
  if s mod 2 = 0 then
    inc(ans,
 cnt[s div 2] * (cnt[s div 2] - 1) div 2);

  assign(f, 'reading.out');
  rewrite(f);
  writeln(f, ans);
  close(f);
end.
```

3. Фізкультура

Розв'язання за допомогою сортування

```
/* GCC */

#define M 200000   // Найбільше можливе
             // значення n

#include <stdio.h>
#include <algorithm>

using namespace std;

int n, i, k;

// Невідсортований і відсортований варіанти
// послідовності відповідно:
int a[M], b[M];

int main()
{
  freopen("pe.in", "r", stdin);
  freopen("pe.out", "w", stdout);

  scanf("%d", &n);
  for (i = 0; i < n; i++)
  {
    scanf("%d", &a[i]);
    b[i] = a[i];
  }

 // Сортування масиву b в порядку
 // незменшення елементів:
  sort(b, b + n);

 // Перевертаємо масив, щоб елементи йшли
 // в порядку незбільшення:
  reverse(b, b + n);

  k = 0;
  for (i = 0; i < n; i++)
   // Коли ми натрапили на черговий
   // елемент з відсортованого масиву,
   // збільшуємо лічильник на одиницю:
    if (a[i] == b[k])
      k++;

  printf("%d\n", n - k);

  return 0;
}
```

Розв'язання за лінійний час

```
/* GCC */

#include <stdio.h>
#include <algorithm>
#include <stack>

using namespace std;
```

```
int n, i, h;
int m;  // Поточне найбільше з чисел,
      // що не входять до стека
stack<int> s;

int main()
{
  freopen("pe.in", "r", stdin);
  freopen("pe.out", "w", stdout);

  scanf("%d", &n);
  m = 0;
  for (i = 0; i < n; i++)
  {
    scanf("%d", &h);
   // Поки верхній елемент стека менший
   // за зчитане число:
    while (! s.empty() && s.top() < h)
    {
     // Оновлюємо максимум чисел,
     // що не входять до стека:
      m = max(m, s.top());

     // І видаляємо зі стека сам
     // верхній елемент:
      s.pop();
    }

   // Якщо зчитане число не менше
   // за поточне значення m, додаємо
   // його в стек:
    if (h >= m)
      s.push(h);
  }

 // Тепер стек містить найдовшу
 // послідовність найвищих учнів, що стоять
 // у порядку незбільшення зросту, тож
 // відповідь — різниця числа n
 // і кількості елементів у стеку:
  printf("%d\n", n - s.size());

  return 0;
}
```

4. Математика

```
/* GCC */

#define M 100  // Найбільша кількість води,
   // що може вміститися в одній посудині

#include <stdio.h>

int v[3];   // У термінах умови задачі
      // v[0] == a, v[1] == b, v[2] == c
int n, ans; // Кількість літрів, яку треба
        // відміряти, і змінна, де буде
        // зберігатись відповідь
int c[3], step; // Стан, який ми
 // розглядаємо на даному кроці, і найменша
 // кількість операцій, за яку даний стан
 // можна дістати з початкового
 // стану (0, 0, 0)
int steps[M + 1][M + 1][M + 1];
 // У steps[x][y][z] буде зберігатися
 // кількість операцій, за яку трійку
 // (x, y, z) можна дістати з початкового
 // стану (0, 0, 0); для станів, які ми
 // поки що не отримали, це значення
 // дорівнює 0

// Розмірності M+1 замість M стоять
// у зв'язку із тим, що кожен елемент
// трійки може набувати одного
// з M+1 значень: від 0 до M включно

int q[(M + 1) * (M + 1) * (M + 1)][3],
 q_start, q_end; // Черга зі станів, які ми
 // дістали з початкової трійки (0, 0, 0)

/* Ініціалізація відповіді, черги станів
та індексного масиву steps */
void init()
{
  ans = 0;

  for (int x = 0; x <= M; x++)
    for (int y = 0; y <= M; y++)
      for (int z = 0; z <= M; z++)
        steps[x][y][z] = 0;

  q_start = q_end = 0;
}

/* Додавання стану (x, y, z) у чергу і до
індексного масиву steps, а також оновлення
відповіді, якщо одне з чисел x, y або z
дорівнює потрібному значенню (числу n);
але якщо стан уже було додано раніше,
то ігноруємо його */
void add(int x, int y, int z)
{
 // Якщо стан уже було отримано на одному
 // з попередніх кроків, то нічого робити
 // не потрібно:
  if (steps[x][y][z] != 0)
    return;

 // Якщо ми вперше дістали рівно n літрів,
 // оновлюємо відповідь:
  if ((x == n || y == n || z == n)
                && ans == 0)
    ans = step;

 // Оновлення індексного масиву (поточна
 // кількість дій зберігається у глобальній
 // змінній step):
  steps[x][y][z] = step;

  q[q_end][0] = x;   // Додаємо стан
  q[q_end][1] = y;   // у чергу
  q[q_end][2] = z;
  q_end++;
}

/* Наповнюємо посудину під номером to
(0 <= to <= 2) до максимального
рівня v[to] */
void fill(int to)
{
  int cc[3]; // Тимчасовий масив, де буде
        // зберігатися отриманий стан

 // Спершу копіюємо в нього поточний стан:
  for (int i = 0; i < 3; i++)
    cc[i] = c[i];
```

```
  cc[to] = v[to]; // Оновлюємо вміст
            // потрібної посудини
  add(cc[0], cc[1], cc[2]); // Додаємо
          // отриманий стан до черги
}

/* Випорожняємо посудину під номером from
(0 <= from <= 2) */
void empty(int from)
{
  int cc[3]; // Тимчасовий масив, де буде
        // зберігатися отриманий стан

 // Спершу копіюємо в нього поточний стан:
  for (int i = 0; i < 3; i++)
    cc[i] = c[i];

  cc[from] = 0; // Оновлюємо вміст
            // потрібної посудини
  add(cc[0], cc[1], cc[2]);  // Додаємо
          // отриманий стан до черги
}

/* Переливаємо воду з посудини from
у посудину to (0 <= from <= 2,
0 <= to <= 2, from != to) */
void pour(int from, int to)
{
  int cc[3]; // Тимчасовий масив, де буде
        // зберігатися отриманий стан

 // Спершу копіюємо в нього поточний стан:
  for (int i = 0; i < 3; i++)
    cc[i] = c[i];

 // Оновлюємо вміст посудини, куди
 // переливають воду:
  cc[to] = c[to] + c[from] > v[to] ?
 /* Якщо в сумі води вистачає,
  щоб її наповнити: */          v[to] :
 /* Якщо води менше: */    c[to] + c[from];

 // Зменшуємо кількість води у посудині,
 // звідки переливали воду, на кількість
 // води, яку вдалося перелити в іншу
 // посудину:
  cc[from] -= cc[to] - c[to];

 // Додаємо отриманий стан до черги:
  add(cc[0], cc[1], cc[2]);
}

int main()
{
  freopen("math.in", "r", stdin);
  freopen("math.out", "w", stdout);

  // Для зручності замість змінних
  // a, b, c використаємо масив v:
  scanf("%d %d %d\n%d",
          &v[0], &v[1], &v[2], &n);

  init();

  step = 0;
  add(0, 0, 0); // Додаємо в чергу
        // початковий стан (0, 0, 0)
// Поки не отримано відповідь
// і черга ще не скінчилася:
  while (ans == 0 && q_end > q_start)
  {
   // Копіюємо в масив с поточний стан
   // із черги:
    for (int i = 0; i < 3; i++)
      c[i] = q[q_start][i];

   // Відразу можемо пересунути вказівник
   // на наступний елемент черги:
    q_start++;

   // Далі пробуємо дістати з поточного
   // стану нові, використовуючи різні
   // альтернативи:

   // Для станів, які буде отримано далі,
   // збільшуємо кількість операцій на 1
   // порівняно з поточним станом:
    step = steps[c[0]][c[1]][c[2]] + 1;

   // Заповнюємо першу, другу,
   // третю посудину:
    fill(0);
    fill(1);
    fill(2);

   // Випорожняємо першу, другу, третю
   // посудину:
    empty(0);
    empty(1);
    empty(2);

   // Переливаємо воду з першої посудини
   // в другу:
    pour(0, 1);

   // Переливаємо воду з другої посудини
   // в першу:
    pour(1, 0);

   // Переливаємо воду з другої посудини
   // в третю:
    pour(1, 2);

   // Переливаємо воду з третьої посудини
   // в другу:
    pour(2, 1);

   // Переливаємо воду з третьої посудини
   // в першу:
    pour(2, 0);

   // Переливаємо воду з першої посудини
   // в третю:
    pour(0, 2);
  }

  // Виводимо відповідь (зокрема 0, якщо
  // черга станів скінчилася раніше,
  // ніж дістали n літрів):
  printf("%d\n", ans);

  return 0;
}
```

(Далі буде)