Департамент образования и молодежной политики
Ханты-Мансийского автономного округа – Югры

Автономное учреждение
дополнительного профессионального образования
Ханты-Мансийского автономного округа – Югры
«Институт развития образования»

А.В. Алексеев

Олимпиады по информатике в Ханты-Мансийском автономном округе – Югре (2008 – 2013 гг.)

Сборник олимпиадных заданий
муниципального этапа
всероссийской олимпиады школьников

Ханты-Мансийск
2013

ББК 74.263.2
А 47

Издается в рамках государственного задания Департамента образования и молодежной политики Ханты-Мансийского автономного округа – Югры. Приказ от 14.12.2012 г. *№ 1479*

Рекомендовано решением Научно-методического совета АУ ДПО ХМАО – Югры «Институт развития образования» Протокол № 4 от 28 мая 2013 г.

Рецензенты:
Семёнов С.П., к.ф.-м.н., доцент, заведующий кафедрой компьютерного моделирования и информационных технологий Югорского государственного университета
Царегородцев А.Л., к.т.н., доцент, заместитель директора Югорского НИИ ИТ

Авторы-составители:
Алексеев А.В., к.п.н., доцент, главный научный сотрудник Югорского НИИ ИТ
Карелин В.А., аспирант Югорского государственного университета
Короткова Е.М. ведущий программист Югорского НИИ ИТ

Алексеев А.В. и др. **Олимпиады по информатике в Ханты-Мансийском автономном округе – Югре (2008 – 2013 гг.):** сборник олимпиадных заданий муниципального этапа всероссийской олимпиады школьников для учащихся 7-11 классов /Вступ. статья А.В. Алексеева /А.В. Алексеев, В.А. Карелин, Е.М. Короткова /Общая редакция Е.Г. Мазуровой. / – Ханты-Мансийск: Редакционно-издательский отдел АУ ДПО ХМАО – Югры «Институт развития образования», 2013.- 108 с.

ISBN 978-5-94611-165-2

В сборнике представлены задания муниципального этапа всероссийской олимпиады школьников по информатике с 2007-2008 по 2012-2013 учебные годы. Для каждой из задач предусмотрены её разбор и программа, используемые знания и примерная сложность. Пособие ориентировано на школьников 7-11 классов, студентов сузов и вузов, учителей информатики общеобразовательных учреждений и преподавателей программирования различных учебных заведений дополнительного и профессионального образования.

ISBN 978-5-94611-165-2

ББК 74.263.2

Введение

В сборнике представлены задания муниципального этапа Всероссийской олимпиады школьников по информатике в 2007-2008 – 2012-2013 учебных годах. Такой выбор обусловлен тем, что в 2007-2008 учебном году было принято Министерством образования и науки РФ новое Положение о проведении всероссийской олимпиады школьников. Также с этого учебного года, впервые в практике проведения такого уровня олимпиад школьников по информатике, была использована автоматизированная проверяющая система сайта в сети интернет «Олимпиады по информатике, ХМАО-Югра» (адрес – http://acmu.ru) [10]. Это позволило всем школьникам автономного округа выполнять единые задания в одно и то же время, что, несомненно, способствовало повышению объективности оценки работы учащихся, сравнению степени их подготовленности к олимпиадам в разных муниципалитетах и учебных заведениях.

Для каждой из 64-х задач проведённых олимпиад приведены её условие, описания входных и выходных данных, примеры, разбор метода или методов её решения, программа на алгоритмическом языке Паскаль. Программы на языке программирования Си, которые подготовлены Е.М. Коротковой, не вошли в печатную версию и могут быть взяты на сайте «Подготовка к олимпиадам в ХМАО-Югре» (адрес – http://zvn.uriit.ru) в разделе, посвященном этому задачнику. Также материалы, вошедшие в задачник, можно взять в разделе «Муниципальные этапы ВсОШ» на сайте Методической службы издательства «БИНОМ. Лаборатория знаний» по адресу http://metodist.lbz.ru/lections/6/.

На основе анализа задач, использовавшихся на олимпиадах, материалов других авторов в сборнике предлагается примерная программа подготовки учащихся к школьному и муниципальному этапу олимпиады школьников по информатике. Сопоставление заданий сборника с аналогичными материалами других регионов России показало, что разработка заданий выполнена очень качественно и эти материалы имеют достаточный для такого уровня соревнований уровень сложности. Сопоставление проводилось на материалах муниципального этапа в г. Москве, Новосибирской области, Республики Карелия, Красноярского и Ставропольского краёв и нескольких других регионов России, где олимпиады также проводятся централизованно

с использованием интернет-сайтов с автоматизированными проверяющими системами.

На сайте «Интернет-олимпиады для школьников ХМАО-Югры» (адрес – http://olymp.uriit.ru) реализована возможность проведения тренировок школьников при подготовке к олимпиадам с использованием всех задач предлагаемого сборника. Интерфейсы участника и преподавателя разработаны В.А. Карелиным и их описание приведено в соответствующем параграфе. Это позволяет преподавателю организовать подготовку своих школьников к олимпиадам в удобное время с выбором набора задач по определённой тематике или сложности.

В тестировании подготовленных материалов и сайта приняли участие учащиеся БУОО Ханты-Мансийского автономного округа – Югры «Югорский физико-математический лицей», МБОУ г. Ханты-Мансийска «Средняя общеобразовательная школа № 1 им. Ю.Г. Созонова», студенты Ханты-Мансийского технолого-педагогического колледжа. Активнее всех работали Николай Павлушин, Александр Субач, Иван Жуков, Никита Сычёв, Евгений Косьмин, Алеся Курбанова. Большое содействие в становлении представленного подхода к проведению олимпиад оказали консультант Департамента образования и молодёжной политики автономного округа Н.В. Гусева, специалисты научно-методического центра Института развития образования по главе с Е.Г. Мазуровой. Авторы пособия всем им выражают искреннюю благодарность.

Муниципальный этап 2007-2008 уч. года, 1-й тур

1. Сумма факториалов

(Время: 1 сек. Память: 16 Мб)

Факториалом натурального числа K называется произведение $K!=1\times2\times3\times\ldots\times K$.

Требуется написать программу, которая по заданному числу N вычислит сумму $1!+2!+\ldots+N!$.

Входные данные

Входной файл input.txt содержит одно натуральное число N ($N \le 200$).

Выходные данные

Выходной файл output.txt должен содержать все десятичные знаки искомой суммы.

Примеры

№	input.txt	output.txt
1	1	1
2	2	3
3	3	9

Разбор

Так как факториал быстро растущая целочисленная функция, то для получения суммы факториалов необходимо использовать длинную арифметику. Если вычислять сумму по формуле слева направо, то потребуется реализовать две операции длинной арифметики: сложение многозначных чисел и умножение многозначного числа на короткое. Для упрощения набора используемых операций преобразуем формулу 1!+2!+…+N!=1+2*(1+3*(…(1+N)…)). А это позволяет использовать вместо сложения более просто программируемую операцию добавления единицы.

Программа

```
var
  n, p, i, k, l : integer;
  a : array [1..1000] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  k:=1; a[k]:=1;
  for i:=n downto 2 do begin p:=0;
    for l:=1 to k do begin
      p:=p+a[l]*i; a[l]:=p mod 10; p:=p div 10 end;
    while p>0 do begin k:=k+1; a[k]:=p mod 10; p:=p
div 10 end;
    l:=1; while a[l]=9 do begin a[l]:=0; l:=l+1 end;
    a[l]:=a[l]+1; if l>k then k:=l
  end;
  for i:=k downto 1 do write(a[i])
end.
```

2. K-удивительные числа

(Время: 3 сек. Память: 16 Мб)

Переворотом числа X назовем число, в котором все цифры числа X стоят в обратном порядке. Например, переворотом числа 6372 является число 2736, а числа 7800 - 87. Назовем K-удивительным такое число, которое в сумме со своим переворотом дает число K.

Например, у числа 222 имеется всего два K-удивительных числа: 111 и 210, а у числа 1050 имеется девять K-удивительных числа: 129, 228, 527, 426, 525, 624, 723, 822, 921.

Требуется написать программу, которая по заданному числу K определит количество K-удивительных чисел.

Входные данные

Входной файл input.txt содержит одно натуральное число K ($1 \le K \le 10^6$).

Выходные данные

Выходной файл output.txt должен содержать одно число – количество K-удивительных чисел.

Примеры

№	input.txt	output.txt
1	222	2
2	1050	9

Разбор

Организуем перебор всех чисел от 1 до заданного числа. Для каждого из этих чисел проверяем его K-удивительность, используя определение из условия задачи.

Программа на Паскале

```
var
  k, i, j, m, s : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  s:=0;
  for i:=1 to k do begin
    j:=i; m:=0;
    while j>0 do begin
      m:=m*10+j mod 10;
      j:=j div 10
    end;
    if i+m=k then s:=s+1
```

```
  end;
  write(s)
end.
```

3. Рамка из клеток

(Время: 1 сек. Память: 16 Мб)

Прямоугольник состоит из X×Y квадратных клеток одинакового размера. Из него вырезан прямоугольник размером (X-2)×(Y-2) так, что осталась рамка шириной в одну клетку. Определить, можно ли покрыть всю рамку плитками размером A×1. Запас плиток неограничен, плитки не накладываются одна на другую и за пределы рамки не выходят.

Требуется написать программу, которая решает эту задачу.

Входные данные

Входной текстовый файл input.txt содержит в первой строке натуральное число K – количество тестов ($1 \le K \le 10$). В следующих K строках записаны по три натуральных числа: X, Y – размеры рамки, A – размер плитки ($3 \le X, Y \le 2\times10^9$, $1 \le A \le 2\times10^9$). Числа разделены пробелами.

Выходные данные

Выходной текстовый файл output.txt должен содержать одну строку из K символов 0 или 1 (1 – если покрытие существует, 0 – иначе).

Примеры

№	input.txt	output.txt
1	1 3 3 1	1
2	2 3 3 2 3 3 3	10

Разбор

Рамку можно покрыть плитками A×1, если A=1 или A=2 или одна из сторон кратна A, а вторая при делении на A дает остаток 2, или каждая при делении на A дает остаток 1.

Программа на Паскале

```
var
  k : integer;
  x, y, a : longint;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(k);
  while k>0 do begin k:=k-1;
    readln(x,y,a);
    if (a<3) or (x mod a=0) and (y mod a=2) or
       (x mod a=2) and (y mod a=0) or
       (x mod a=1) and (y mod a=1) then write(1) else
write(0)
  end
end.
```

Муниципальный этап 2007-2008 уч. года, 2-й тур

4. Подарки Деда Мороза

(Время - 1 сек., память - 16 Мб)

Ириска весит X грамм, мандарин – Y грамм, пряник – Z грамм.

Требуется написать программу, которая определит, сколько различных вариантов подарков весом ровно W грамм может сделать Дед Мороз.

Входные данные

В единственной строке входного файла input.txt содержатся четыре числа X, Y, Z и W ($1 \le X, Y, Z \le 100$, $1 \le W \le 1000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число – количество вариантов подарков.

Пример

№	input.txt	output.txt
1	10 25 15 40	3

Примечание

В приведенном примере следующие варианты: 4 ириски, 1 мандарин и 1 пряник, 1 ириска и 2 пряника.

Разбор

Математически задача сводится к нахождению количества неотрицательных решений (a, b, c) уравнения $a*x+b*y+c*z=w$. Если организовать тройной цикл и проверять выполнение уравнения, то, например, при $x=y=z=1$ и $w=1000$ понадобится 10^9 проверок. Организуем двойной цикл, например, по переменным a и b и будем проверять разрешимость уравнения относительно c.

Программа на Паскале

```
var
  x, y, z, w, a, b, v : integer;
  k : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x,y,z,w);
  k:=0;
  for a:=0 to w div x do
    for b:=0 to w div y do
    begin
      v:=w-a*x-b*y;
      if (v>=0) and (v mod z=0) then k:=k+1
    end;
  write(k)
end.
```

5. Следующее число

(Время - 1 сек., память - 16 Мб)

Задано натуральное число N.

Требуется написать программу, которая найдет следующее за ним число, в двоичном разложении которого столько же единиц, сколько в двоичном разложении числа N.

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число N ($1 \le N \le 2^{30}$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – следующее за заданным, в двоичном разложении которого столько же единиц.

Примеры

№	input.txt	output.txt
1	1	2
2	2	4
3	3	5

Разбор

Найдем двоичное разложение заданного числа. Просматривая его слева направо, пропускаем нули, первую единицу заменяем нулем, также далее следующую группу единиц заменяем нулями, подсчиты-

вая количество таких замен, первый встретившийся ноль заменяем на единицу, справа дописываем подсчитанное количество единиц.

Программа на Паскале

```
var
  n : longint;
  i, k, j : integer;
  a : array [1..32] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); k:=0;
  while n>0 do begin k:=k+1; a[k]:=n mod 2; n:=n div
2 end;
  i:=1; while a[i]=0 do i:=i+1;
  a[i]:=0; i:=i+1; j:=0;
  while a[i]=1 do begin a[i]:=0; i:=i+1; j:=j+1 end;
  a[i]:=1; if i>k then k:=i;
  for i:=1 to j do a[i]:=1;
  if k=32 then write('2147483648') else
  begin
    for i:=k downto 1 do n:=n*2+a[i];
    write(n)
  end
end.
```

6. Точки отрезка

(Время - 1 сек., память - 16 Мб)

Концы отрезка на плоскости имеют целочисленные координаты.

Требуется написать программу, которая вычислит, сколько всего точек с целочисленными координатами принадлежат этому отрезку.

Входные данные

В единственной строке входного файла input.txt записаны четыре числа – координаты концов отрезка (x_1, y_1) и (x_2, y_2). Каждая из координат не превышает по абсолютной величине значения 10^9.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – количество точек на заданном отрезке, имеющих целочисленные координаты.

Примеры

№	input.txt	output.txt
1	1 1 2 2	2
2	0 0 -2 -2	3
3	1 1 1 10	10

Разбор

Пусть (x_1, y_1), (x_2, y_2) – координаты концов отрезка. Переместим отрезок и, если нужно, отразим его относительно вертикали и горизонтали так, чтобы его левый нижний конец находился в точке (0, 0), а второй имел координаты $(x, y)=(|x_1-x_2|, |y_1-y_2|)$. Очевидно, что количество «целочисленных» точек на отрезке при этих перемещениях не изменяется. Найдем наибольший общий делитель координат отрезка, d=НОД(x, y). Тогда x=k·d, y=l·d, то точки (k, l), (2k, 2l), …,((d-1)·k, (d-1)·l) принадлежат отрезку, поэтому всего «целочисленных» точек (d-1)+2=d+1.

Программа на Паскале

```
var
  x1, y1, x2, y2, a, b, c : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(x1,y1,x2,y2); a:=abs(x1-x2); b:=abs(y1-y2);
  if a=0 then write(b+1) else
    if b=0 then write(a+1) else begin
      while b>0 do
      begin c:=a mod b; a:=b; b:=c end;
      write(a+1)
    end
end.
```

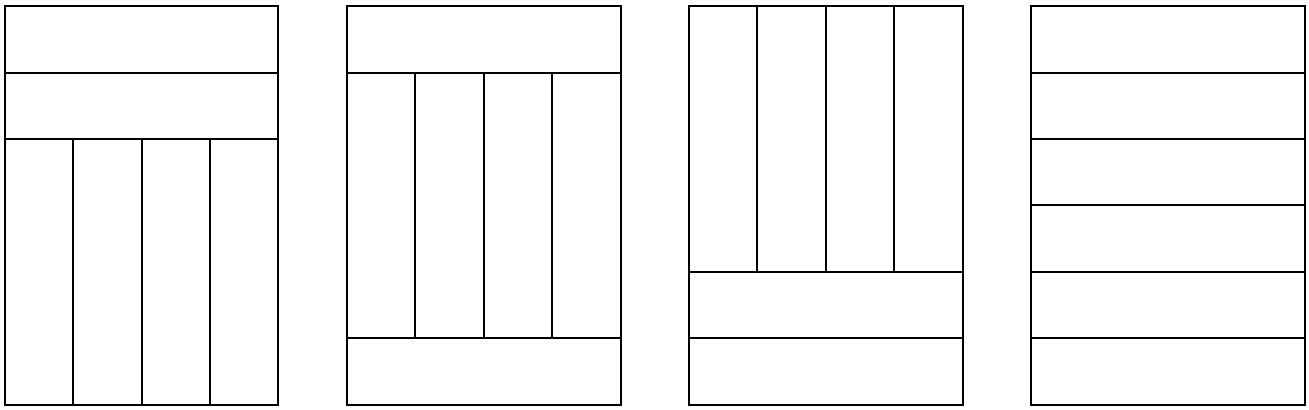
Муниципальный этап 2007-2008 уч. года, 3-й тур

7. Коридор

(Время - 1 сек., память - 16 Мб)

Прямоугольный коридор длиной N метров и шириной M метров решили застелить N прямоугольными плитками шириной 1 метр и длиной M метров, таким образом, чтобы не было не застеленной поверхности.

Требуется написать программу, которая найдет количество способов это сделать. Например, для коридора с размерами 6 на 4 существует четыре способа застелить плитками 1 на 4.



Входные данные

В единственной строке входного файла input.txt записаны два целых числа – M (длина плитки и ширина коридора) и N (длина коридора). Для этих чисел верны неравенства $2 \le M \le N \le 50$.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – количество способов.

Примеры

№	input.txt	output.txt
1	4 6	4
2	2 2	2

Разбор

Задача решается методом динамического программирования. Для этого обозначим через A(n) – количество способов замостить коридор длиной n. Так как для коридора длиной k<m плитки можно укладывать только горизонтально, то A(k)=1. В других случаях плитки в последнем ряду можно положить или горизонтально или вертикально, а тогда A(n)=A(n-1)+A(n-m). Приведенные соотношения позволяют последовательно рассчитать требуемое количество замощений коридора. Ответом будет значение A(N).

Программа на Паскале

```
var
  m, n, i : integer;
  a : array [0..50] of int64;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(m,n);
  for i:=0 to m-1 do a[i]:=1;
  for i:=m to n do
    a[i]:=a[i-1]+a[i-m];
  write(a[n])
end.
```

8. Различные цифры

(Время - 1 сек., память - 16 Мб)

Дано целое число N.

Требуется написать программу, определяющую, в каких системах счисления с основаниями от 2 до 36 это число не содержит одинаковых цифр.

Входные данные

В единственной строке входного файла input.txt записано одно целое число N ($1 \le N \le 10^9$), представленное в десятичной системе счисления.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести основания систем счисления в порядке возрастания, разделенные одним пробелом.

Пример

№	input.txt	output.txt
1	100	11 12 13 14 15 16 17 18 20 21 22 23 25 26 27 28 29 30 31 32 33 34 35 36

Разбор

В цикле от 2 до 36 по основанию системы счисления находим для каждой из цифр этой системы счисления ее количество в записи числа. Далее проверяем требуемое условие: в записи числа нет одинаковых цифр.

Программа на Паскале

```
var
  n, i : longint;
  k, j, m : integer;
  c : array [0..35] of integer;
  t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for k:=2 to 36 do
  begin
    i:=n; for j:=0 to k-1 do c[j]:=0;
    while i>0 do
    begin
      m:=i mod k; c[m]:=c[m]+1; i:=i div k
    end;
    t:=true;
    for j:=0 to k-1 do t:=t and (c[j]<2);
```

```
    if t then write(k,' ')
  end
end.
```

9. Слово

(Время - 1 сек., память - 16 Мб)

Числа Фибоначчи строятся следующим образом: первые два равны единице, а каждое следующее равно сумме двух предыдущих. Например, первые десять чисел Фибоначчи равны: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55. В заданном тексте символы нумеруются слева направо, начиная с единицы.

Требуется написать программу, которая составит слово из символов, номера которых совпадают с числами Фибоначчи.

Входные данные

В единственной строке входного файла input.txt записан текст, состоящий из латинских строчных букв. В тексте не более 30000 символов.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести слово из символов, номера которых совпадают с числами Фибоначчи. Символы слова идут в том же порядке, что и в заданном тексте.

Примеры

№	input.txt	output.txt
1	a	a
2	abc	abc
3	abcdefghij	abceh

Разбор

Вводим символы из файла, одновременно подсчитывая их номера и числа Фибоначчи. Если номер введенного символа совпадает с очередным числом Фибоначчи, то выводим этот символ.

Программа на Паскале

```
var
  s : char;
  a, b, c, i : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  a:=1; b:=1; i:=0;
  while not eoln do
  begin
```

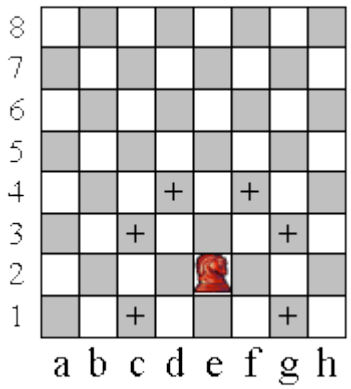
```
    i:=i+1;
    read(s);
    if i=b then
    begin write(s);
      c:=a+b; a:=b; b:=c
    end
  end
end.
```

Муниципальный этап 2008-2009 уч. года, 1-й тур

10. Шахматный конь

(Время: 1 сек. Память: 16 Мб)

Вася решил научиться играть в шахматы. Он нашел книгу с записями партий и внимательно их изучает. Может быть, когда-нибудь Вася станет великим шахматистом, но пока он еще учится в начальной школе, и ему нелегко дается шахматная нотация. Больше всего трудностей у Васи вызывают ходы шахматного коня. Он попросил вас написать программу, которая сможет сообщить Васе, на какие клетки можно пойти конем с заданной клетки.



Вы, наверное, тоже знаете, что конь в шахматах всегда перемещается либо на две клетки по горизонтали и на одну по вертикали, либо на одну по горизонтали и на две по вертикали. Вертикали обозначаются маленькими латинскими буквами от a до h, а горизонтали - цифрами от 1 до 8. Любая клетка на шахматной доске обозначается буквой соответствующей вертикали и цифрой соответствующей горизонтали, например, c6 или e2.

Входные данные

Во входном файле input.txt записано 2 символа – координаты клетки, где стоит конь.

Выходные данные

В выходной файл output.txt в произвольном порядке выведите все координаты клеток, на которые за один ход может попасть конь, находящийся на заданной клетке.

Примеры

№	input.txt	output.txt
1	e2	c1 c3 d4 f4 g1 g3
2	a1	b3 c2

Разбор

Переведём координаты заданной клетки из шахматной нотации в математическую. После этого просмотрим все поля шахматной доски и определим те из них, в которые может попасть конь с заданной клетки.

Программа на Паскале

```
var
  s : string;
  i, j, k, l : integer;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(s);
  k:=ord(s[1])+1-ord('a'); l:=ord(s[2])-48;
  for i:=1 to 8 do
    for j:=1 to 8 do
      if (abs(k-i)=2)and(abs(l-j)=1) or
         (abs(k-i)=1)and(abs(l-j)=2) then
           writeln(chr(i-1+ord('a'))+chr(j+48));
  close(output)
end.
```

11. Химическая формула

(Время: 1 сек. Память: 16 Мб)

Запись химической реакции всегда содержит описания нескольких веществ. В свою очередь, описание одного химического вещества – строка, в которой входящие в него атомы химических элементов перечисляются в определенном порядке. При этом последовательности из двух и более одинаковых атомов, идущих подряд, группируются: записывается сокращенное название химического элемента и количество одинаковых элементов подряд. Например, вместо HH пишут H2. Обозначения химических элементов состоят из одной или

двух латинских букв, из которых первая - прописная, а вторая – строчная. В этой задаче не будут рассматриваться более сложные правила. Например, не используются скобки. Вы должны проверить, что заданная последовательность символов подходит под данное выше описание формулы химического вещества. При этом не нужно рассматривать корректность заданной строки, исходя из каких-либо других соображений, даже если они продиктованы здравым смыслом.

Входные данные

В единственной строке входного файла input.txt записана последовательность символов, содержащая только цифры и строчные и прописные латинские буквы. Гарантируется, что в последовательности перед каждой строчной буквой идет прописная, а все однобуквенные и двухбуквенные подстроки, начинающиеся с прописной буквы – правильные обозначения химических элементов (поэтому здесь даже не приводится их список). Длина последовательности не превосходит 1000 символов.

Выходные данные

В выходной файл output.txt выведите одно слово YES, если данная строка подходит под упрощенное описание формулы химического вещества из условия и NO, если не подходит.

Примеры

№	input.txt	output.txt
1	OHNaOHNa	YES
2	H2O	YES
3	HH	NO
4	CHC	YES

Разбор

Последовательно обработаем символы введённой строки. При определении нарушения представленного описания химической формулы печатаем соответствующее сообщение и завершаем работу программы. В нижеприведённой программе это сделано с помощью подпрограммы. Если была просмотрена вся строка, то печатаем сообщение о правильности записи химической формулы.

Программа на Паскале

```
procedure quit(yes: boolean);
begin
  if (yes) then writeln('YES')
           else writeln('NO');
  halt
end;
```

```
var
  s, cur, prev: string;
  i, j, n: integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s); s:=s+'9'; n:=length(s);
  i:=1; prev:='';
  while (i<=n-1) do begin
    if not(('A'<=s[i])and(s[i]<='Z')) then quit(false);
    cur:=s[i];
    inc(i);
    if ('a'<=s[i]) and (s[i]<='z') then begin
      cur:=cur+s[i]; inc(i);
    end;
    if (s[i]='0') then quit(false);
    j:=1; if s[i]='1' then j:=-1;
    if (cur=prev) then quit(false);
    while ('0'<=s[i])and(s[i]<='9') and (i<=n-1) do begin
      inc(i); inc(j);
    end;
    if (j<=0) then quit(false);
    prev:=cur;
  end;
  quit(true);
end.
```

12. Время прибытия

(Время: 1 сек. Память: 16 Мб)

Задано время отправления поезда и время в пути до конечной станции. Требуется написать программу, которая найдет время прибытия этого поезда (возможно, в другие сутки).

Входные данные

Входной файл input.txt содержит две строки. В первой строке задано время отправления, а во второй строке – время в пути. Время отправления задается в формате «HH:MM», где HH время в часах, которое принимает значение от 00 до 23, MM – время в минутах, которое принимает значение от 00 до 59. Время в пути задается двумя неотрицательными целыми числами – количество часов и количество минут. Числа разделяются одним пробелом. Количество часов не превышает 120, минут – 59.

Выходные данные

Выходной файл output.txt должен содержать одну строку – время прибытия поезда на конечную станцию. Формат вывода этого времени совпадает с форматом ввода времени отправления.

Примеры

№	input.txt	output.txt
1	00:00 10 10	10:10
2	01:02 4 6	05:08
3	11:00 22 0	09:00

Разбор

Простая задача на работу с данными, задающими время (часы, минуты). Требуется правильно осуществить ввод и применить операции целочисленной арифметики.

Программа на Паскале

```
var
  ho, mo, hp, mp : integer;
  s : string;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  ho:=(ord(s[1])-48)*10+ord(s[2])-48;
  mo:=(ord(s[4])-48)*10+ord(s[5])-48;
  read(hp,mp);
  hp:=ho+hp; mp:=mo+mp;
  if mp>59 then begin mp:=mp-60; hp:=hp+1 end;
  while hp>23 do hp:=hp-24;
  if hp<10 then write(0); write(hp,':');
  if mp<10 then write(0); write(mp)
end.
```

Муниципальный этап 2008-2009 уч. года, 2-й тур

13. Постоянная Капрекара

(Время: 1 сек. Память: 16 Мб)

Возьмем четырехзначное число, в котором не все цифры одинаковы, например 6264. Расположим цифры сначала в порядке убывания - 6642; затем, переставив их в обратном порядке, получим 2466. Вычтем последнее число из 6642. На следующем шаге с полученной

разностью проделаем то же самое. Через несколько таких действий получится число, переходящее само в себя и называемое постоянной Капрекара.

Требуется написать программу, которая находит эту постоянную и количество шагов для ее получения из заданного четырехзначного числа.

Входные данные

Входной файл input.txt содержит одну строку, в которой записано четырехзначное число.

Выходные данные

В выходной файл output.txt записываются: в первой строке постоянная Капрекара, во второй – количество шагов для ее получения.

Примеры

№	input.txt	output.txt
1	1234	6174 3

Разбор

Проводя последовательные вычисления по описанному правилу, останавливаемся при обнаружении совпадения двух соседних чисел. Ответом будет уменьшенное на единицу значения счётчика количества выполняемых вычислений.

Программа

```
var
  n, n1, c : integer;
function sled(n:integer): integer;
  var a : array [1..4] of integer;
      i, j, x : integer;
begin
  for i:=1 to 4 do
  begin
    a[i]:=n mod 10;
    n:=n div 10
  end;
  for i:=1 to 3 do
    for j:=1 to 4-i do
      if a[j]>a[j+1] then begin x:=a[j]; a[j]:=a[j+1];
a[j+1]:=x end;
  sled:=(a[4]-a[1])*999+(a[3]-a[2])*90
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
```

```
  read(n);
  c:=0; n1:=sled(n);
  while n<>n1 do
  begin
    c:=c+1;
    n:=n1;
    n1:=sled(n)
  end;
  writeln(n);
  write(c)
end.
```

14. Короткая последовательность

(Время: 1 сек. Память: 16 Мб)

Дано целое число N. Рассмотрим последовательность $S_1S_2S_3...S_k...$, где каждая группа цифр S_k состоит из записанных одно за другим чисел от 1 до k. Например, первые 75 цифр последовательности выглядят так:

112123123412345123456123456712345678123456789123456789101234567891011123456.

Требуется написать программу, которая определит цифру на N-ой позиции построенной последовательности.

Входные данные

Входной файл input.txt содержит одно число N ($0 < N < 32768$).

Выходные данные

В выходной файл output.txt выведите цифру, которая стоит на N-ой позиции в последовательности.

Примеры

№	input.txt	output.txt
1	3	2
2	20	5

Разбор

Вначале найдем число, которое содержит указанную позицию. Затем уже определяем цифру.

Программа

```
var
  n, k, l, m : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  l:=0; k:=0;
```

```
  while n>l+m do
  begin
    k:=k+1;
    if k<10 then m:=1 else
      if k<100 then m:=2 else m:=3;
    l:=l+m;
    n:=n-l
  end;
  if n<10 then write(n) else
    if n<190 then
    begin
      n:=n-10; l:=n div 2+10;
      if n mod 2=0 then write(l div 10) else write(l mod
10)
    end
    else
    begin
      n:=n-190; l:=n div 3+100;
      case n mod 3 of
        0 : write(l div 100);
        1 : write(l div 10 mod 10);
        2 : write(l mod 10)
      end
    end
end.
```

15. Последовательности из 0 и 1

(Время: 1 сек. Память: 16 Мб)

Рассмотрим последовательности длины N, состоящие из 0 и 1. Требуется написать программу, которая по заданному натуральному числу N определяет количество тех из них, в которых никакие две единицы не стоят рядом.

Входные данные

Входной файл input.txt содержит число N ($1 \le N \le 1000$).

Выходные данные

В выходной файл output.txt выведите ответ на задачу.

Примеры

№	input.txt	output.txt
1	2	3
2	3	5

Разбор

Задача решается методом динамического программирования. Легко заметить, что в результате этого получаются числа Фибоначчи.

Ограничения в задаче требуют для их вычисления применения «длинной» арифметики.

Программа

```
var
  n,i,j,k1,k2,k3,p : integer;
  f1, f2, f3 : array [1..1000] of integer;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n);
  if n=1 then write(2) else
    if n=2 then write(3) else begin
      f1[1]:=2; k1:=1; f2[1]:=3; k2:=1;
      for i:=3 to n do begin
        p:=0; k3:=k2;
        for j:=1 to k1 do begin
          p:=p+f1[j]+f2[j]; f3[j]:=p mod 10; p:=p div 10
end;
        for j:=k1+1 to k2 do begin
          p:=p+f2[j]; f3[j]:=p mod 10; p:=p div 10 end;
        if p>0 then begin k3:=k3+1; f3[k3]:=p end;
        k1:=k2; f1:=f2; k2:=k3; f2:=f3
      end;
      for j:=k2 downto 1 do write(f3[j]);
    end;
  close(output)
end.
```

Муниципальный этап 2008-2009 уч. года, 3-й тур

16. Пасьянс старухи Шапокляк

(Время: 1 сек. Память: 16 Мб)

На столе лежат колоды игральных карт. В самой тоненькой колоде – p карт, во второй – p+1, в третьей – p+2, …, в последней – k карт. Старуха Шапокляк раскладывает пасьянс. Беря в руки любую из колод, она, если число карт в ней четное, на место возвращает колоду, наполовину уменьшив число карт в ней (лишние убирает в ящик), а если количество карт в колоде нечетное, то утраивает их количество и добавляет еще одну карту, а уже тогда кладет колоду на стол (карт у нее в ящике для этой операции достаточно). Если в какой-то колоде остается две карты, она больше ее не трогает. Пасьянс сходится, если во всех колодах остается по две карты.

Требуется написать программу, которая определит сходится ли пасьянс, и если сходится – сколько раз должна старуха Шапокляк брать со стола карты.

Входные данные

Входной файл input.txt содержит 2 числа, записанные через пробел ($2 < p < k < 1000$).

Выходные данные

Выходной файл output.txt должен содержать 0, если пасьянс не сходится, и, если сходится, количество «ходов» старухи Шапокляк.

Примеры

№	input.txt	output.txt
1	2 3	6
2	5 8	28

Разбор

Задача сводится к математическому моделированию описанного в условии задачи действия для каждой из колод. Сходимость пасьянса не доказана математически, но для имеющихся в задаче ограничений проверена экспериментально.

Программа

```
var
  p, k : integer;
  s, t : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(p,k);
  s:=0;
  while p<=k do
  begin
    t:=p;
    while t>2 do
    begin
      s:=s+1;
      if odd(t) then t:=3*t+1 else t:=t div 2;
    end;
    p:=p+1
  end;
  write(s);
end.
```

17. 2-простое число

(Время: 1 сек. Память: 16 Мб)

Число называется 2-простым, если являются простыми числа, составленные из цифр этого числа в возрастающем и убывающем порядках.

Требуется написать программу, которая по заданному числу определит его 2-простоту.

Входные данные

Входной файл input.txt содержит натуральное число N ($10 < N < 10^9$).

Выходные данные

В выходной файл output.txt выведите сообщение «Yes», если число N является 2 – простым и «No» – иначе.

Примеры

№	input.txt	output.txt
1	13	Yes
2	23	No

Разбор

Найдём цифры числа и отсортируем их в порядке возрастания. Для каждого из указанных в условии чисел проверим их простоту. В программе предложено это сделать с помощью логической подпрограммы.

Программа

```
var
  n, m : longint;
  k, i, j, x : integer;
  a : array [1..9] of integer;
function pr(n:longint):boolean;
  var b : boolean; t : longint;
begin
  b:=true;
  if n mod 2=0 then b:=false;
  t:=3;
  while b and (t*t<=n) do
  begin
    if n mod t=0 then b:=false;
    t:=t+2
  end;
  pr:=b
end;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(n);
  k:=0;
  while n>0 do
  begin
    k:=k+1;
    a[k]:=n mod 10;
    n:=n div 10
  end;6-
  for i:=1 to k-1 do
    for j:=1 to k-i do
      if a[j]>a[j+1] then
      begin x:=a[j]; a[j]:=a[j+1]; a[j+1]:=x end;
  for i:=1 to k do n:=n*10+a[i];
  m:=0; for i:=k downto 1 do m:=m*10+a[i];
  if pr(n) and pr(m) then write('Yes') else write('No');
end.
```

18. Арифметическая прогрессия

(Время: 1 сек. Память: 16 Мб)

Заданы первый и второй элементы арифметической прогрессии. Требуется написать программу, которая вычислит элемент прогрессии по ее номеру.

Входные данные

В единственной строке входного файла input.txt записаны три целых числа, разделенных пробелами – первый элемент прогрессии A_1 ($1 \le A_1 \le 1000$), второй элемент прогрессии A_2 ($1 \le A_2 \le 1000$) и номер требуемого элемента N ($1 \le N \le 1000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число – N-й элемент арифметической прогрессии.

Пример

№	input.txt	output.txt
1	1 5 3	9

Разбор

Для решения задачи необходимо использовать формулу для *n*-го члена арифметической прогрессии $a_n=a_1+d\cdot(n-1)=a_1+(a_2-a_1)\cdot(n-1)$.

Программа

```
var
  a1, a2, n : integer;
  an : longint;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(a1,a2,n);
  an:=a2-a1; an:=a1+an*(n-1);
  write(an)
end.
```

Муниципальный этап 2008-2009 уч. года, 4-й тур

19. Уравнение по основанию

(Время: 1 сек. Память: 16 Мб)

Запись A_X обозначает, что A есть запись числа в системе счисления по основанию X. Если X>10, то для записи числа используются кроме цифр от 0 до 9 заглавные латинские буквы от A до Z. При этом условии X не может быть больше 36.

Требуется написать программу, которая по заданным значениям A и B найдет решение уравнения $A_X = B$, либо сообщит об отсутствии у него решений.

Входные данные

Входной файл input.txt содержит в первой строке число A, во второй число B ($1 \le B \le 10^7$).

Выходные данные

В выходной файл output.txt выведите либо основание системы счисления, удовлетворяющее уравнению (если таких несколько, то наименьшее), либо 0, если уравнение не имеет решений.

Примеры

№	input.txt	output.txt
1	A1 161	16
2	201 26	0

Разбор

В начале по записи числа A определим минимально возможное основание системы счисления. После этого, перебором по основаниям системы счисления проверяем выполнение уравнения. Для вычисления значения числа по его цифрам и основанию системы счисления используем схему Горнера.

Программа

```
var
  a, c : string;
  b, z : longint;
  x, l, i : integer;
```

```
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(a); read(b);
  c:='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  x:=0;
  for i:=1 to length(a) do
    if pos(a[i],c)>x then x:=pos(a[i],c);
  z:=0;
  while z<b do
  begin
    z:=0;
    for i:=1 to length(a) do
      z:=z*x+pos(a[i],c)-1;
    if z=b then write(x);
    x:=x+1
  end;
  if z>b then write(0)
end.
```

20. Боулинг

(Время: 1 сек. Память: 16 Мб)

Цель при игре в боулинг – сбить шаром максимальное количество кеглей. Партия в этой игре состоит из 10 туров. Задача игрока – сбить все 10 кеглей в каждом туре. Для этого игрок может совершить 2 броска шара, за исключением:

- если 10 кеглей сбиты первым броском, то второй бросок не совершается;
- если 10 кеглей сбиты первым броском в десятом туре, то игроку предоставляются два призовых броска, а если двумя бросками – один.

Количество очков в каждом туре равно количеству сбитых кеглей, кроме двух бросков, называемых «Strike» и «Spire».

Strike: игрок сбивает 10 кеглей первым броском, очки в этом туре начисляются из расчета – 10 + сумма очков за два последующих броска.

Spire: игрок сбивает 10 кеглей двумя бросками, очки в этом туре начисляются из расчета – 10 + сумма очков за один последующий бросок.

Результат партии складывается из результатов всех 10 туров.

Требуется написать программу, которая определит количество набранных игроком очков.

Входные данные

Входной файл input.txt содержит в первой строке одно натуральное число, определяющее количество совершенных бросков. Вторая строка содержит натуральные числа (разделенные пробелом), обозначающие количество сбитых кеглей за каждый совершенный бросок.

Выходные данные

Выходной файл output.txt должен содержать одно целое число – количество набранных игроком очков.

Примеры

№	input.txt	output.txt
1	12 10 10 10 10 10 10 10 10 10 10 10 10	300
2	20 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0
3	15 10 10 10 8 2 10 3 4 8 2 4 5 10 4 5	173

Разбор

В начале прочитаем данные в массив. Далее циклом по турам подсчитаем набранные игроком очки.

Программа

```
var
  n, i, c, d, s : integer;
  a : array [1..21] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for i:=1 to n do read(a[i]);
  n:=0; s:=0;
  for i:=1 to 10 do
  begin
     n:=n+1; c:=a[n];
     if c=10 then s:=s+10+a[n+1]+a[n+2] else
     begin
        n:=n+1; d:=a[n];
        if c+d=10 then s:=s+10+a[n+1] else s:=s+c+d
     end
  end;
  write(s)
end.
```

21. Земельный комитет

(Время: 1 сек. Память: 16 Мб)

Земельный комитет города принял решение о сдаче в аренду части муниципальной территории, имеющей форму прямоугольника размером H на W километров. Стоимость аренды каждого квадратного участка 1 x 1 км была определена с учётом локальных условий, и занесена в таблицу.

С целью организации открытого тендера на аренду, земельный комитет решил выставить на своём веб-сайте карту территории, и предоставить посетителям возможность узнавать суммарную стоимость аренды для произвольной прямоугольной группы соседних участков.

Данное предложение вызвало большой интерес у населения и предпринимателей, и нагрузка на сервер очень высока.

Требуется написать программу, позволяющую как можно более эффективно рассчитывать стоимость аренды для N запросов. В каждом запросе требуется определить общую стоимость участков внутри прямоугольной группы с противоположными углами, расположенными в элементах таблицы (ai, bi) и (ci, di).

Входные данные

В первой строке входного файла input.txt находятся числа H, W, N ($1 \le H, W \le 100$, $1 \le N \le 1000000$). В следующих H строках содержится по W чисел (стоимости участков находятся в диапазоне от 0 до 10000). Далее идут N строк с числами ai, bi, ci, di ($1 \le ai \le ci \le H$, $1 \le bi \le di \le W$).

Выходные данные

В выходной файл output.txt должен содержать N чисел, по одному числу в строке.

Пример

№	input.txt	output.txt
1	2 3 1 5 1 2 6 7 3 2 1 2 3	16

Разбор

Задача решается методом динамического программирования. Для этого по мере ввода заданной таблицы стоимостей отдельных участков будем насчитывать массив стоимостей аренды прямоугольной группы участков от левого верхнего угла. Это позволяет быстро вы-

числять стоимости требуемых посетителям прямоугольных групп соседних участков.

Программа

```
var
  s : array[0..100,0..100] of longint;
  h, w, i, j, a, b, c, d : integer;
  n, k, x : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(h,w,n);
  for i:=0 to w do s[i,0]:=0;
  for j:=0 to h do s[0,j]:=0;
  for j:=1 to h do
    for i:=1 to w do
    begin
      read(x);
      s[i,j]:=x+s[i-1,j]+s[i,j-1]-s[i-1,j-1]
    end;
  for k:=1 to n do
  begin
    read(a,b,c,d);
    write(s[d,c]-s[d,a-1]-s[b-1,c]+s[b-1,a-1]);
    if k<n then writeln
  end;
end.
```

Муниципальный этап 2008-2009 уч. года, 5-й тур

22. Оптовая покупка

(Время: 1 сек. Память: 16 Мб)

Пара носков стоит 10 руб. 50 коп., связка (12 пар) стоит 102 руб. 50 коп., а коробка (12 связок) стоит 1140 руб.

Требуется написать программу, которая по числу пар носков, которые хочет купить покупатель, вычисляет количества коробок, связок и пар носков, которые ему следует купить с наибольшей выгодой.

Входные данные

Входной файл input.txt содержит натуральное число N ($N \le 10^9$) – число пар носков, которые желает купить покупатель.

Выходные данные

Выходной файл output.txt должен содержать три числа (первое – количество коробок, второе – связок, третье – пар носков), разделенные пробелами.

Примеры

№	input.txt	output.txt
1	11	0 1 0
2	500	3 5 8

Разбор

Обозначим через n1, n2, n3 количество коробок, связок и пар носков, которые следует купить. Оптимальная покупка без излишков находится очевидным образом. Это записано в первых четырех операторах после ввода в нижеприведенной программе. Удешевить покупку можно лишь двумя способами – либо взять лишнюю связку и не брать пар, либо взять лишнюю коробку и не брать ни связок, ни пар.

Программа

```
var
  n, m, n1, n2, n3 : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  n1:=n div 144;
  m:=n mod 144;
  n2:=m div 12;
  n3:= m mod 12;
  if n3*105>1025 then
    begin n2:=n2+1; n3:=0 end;
  if n2*1025+n3*105>11400 then
    begin n1:=n1+1; n2:=0; n3:=0 end;
  write(n1,' ',n2,' ',n3)
end.
```

23. Похожие массивы

(Время: 1 сек. Память: 16 Мб)

Два массива называются похожими, если совпадают множества чисел, встречающихся в этих массивах.

Требуется написать программу, которая определит: похожи ли два заданных массива.

Входные данные

Входной файл input.txt содержит в первой строке два числа M и N - длины массивов ($1 \le M, N \le 16000$). Во второй строке записаны M чисел – элементы первого массива. В третьей строке записаны N чисел – элементы второго массива. Числа в строках разделены пробела-

ми, элементы массивов не превышают по абсолютной величине 32000.

Выходные данные

Выходной файл output.txt должен содержать 1, если массивы похожи и 0 иначе.

Примеры

№	input.txt	output.txt
1	4 3 1 2 3 2 1 2 3	1
2	2 3 1 2 2 3 1	0

Разбор

Отсортируем массивы по возрастанию. В нижеприведенной программе это сделано методом «пузырька». Для решения, удовлетворяющего поставленным в задаче временным ограничениям, требуется более быстрый метод, например, быстрая сортировка Хоара. Далее одновременно просматриваем отсортированные массивы и они не будут похожими в двух случаях: либо найдутся различные элементы, либо один из массивов останется не просмотренным.

Программа

```
var
  m, n, i, j, s : integer;
  a, b : array [1..16000] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(m,n);
  for i:=1 to m do read(a[i]);
  for j:=1 to n do read(b[j]);
  for i:=1 to m-1 do
    for j:=1 to m-i do
      if a[j]>a[j+1] then begin s:=a[j]; a[j]:=a[j+1];
a[j+1]:=s end;
  for i:=1 to n-1 do
    for j:=1 to n-i do
      if b[j]>b[j+1] then begin s:=b[j]; b[j]:=b[j+1];
b[j+1]:=s end;
  i:=1; j:=1;
  while (i<=m) and (j<=n) and (a[i]=b[j]) do
  begin
```

```
    s:=a[i];
    while (i<=m) and (a[i]=s) do i:=i+1;
    while (j<=n) and (b[j]=s) do j:=j+1;
  end;
  if (i>m) and (j>n) then write(1) else write(0)
end.
```

24. Роман в томах

(Время: 1 сек. Память: 16 Мб)

В романе N глав. В i-той главе a_i страниц. Требуется издать роман в K томах так, чтобы объем самого «толстого» тома был минимален. В каждом томе главы располагаются по порядку своих номеров.

Требуется написать программу, которая найдет количество страниц в самом «толстом» томе.

Входные данные

Входной текстовый файл input.txt содержит в первой строке число N – количество глав в романе ($1 \le N \le 100$). Во второй строке через пробел записаны N чисел – количество страниц в каждой главе. Количество страниц в романе не превышает 32767. В третьей строке записано число K – количество томов ($1 \le K \le N$).

Выходные данные

Выходной файл output.txt должен содержать количество страниц в самом «толстом» томе.

Примеры

№	input.txt	output.txt
1	3 1 2 1 2	3
2	4 1 2 1 1 3	2

Разбор

Задача решается методом динамического программирования. Обозначим через $B(i, j)$ – количество страниц в самом «толстом» томе из i глав, если их издавать в j томах по требуемому в условии задачи условию. Получим рекуррентные соотношения на эту величину. Если j=1, то $B(i,1) = \sum_{k=1}^{i} a_k$. Для издания романа в j+1 томе в последний том могут попасть главы: либо i (a_i страниц), либо i-1 и i (a_{i-1} и a_i стра-

ниц), …, либо j+1, j+2, …, i ($\sum_{l=j+1}^{i} a_l$ страниц). Для каждого из этих способов размещения глав в j+1 томе величина B для j томов уже вычислена. Берем максимум из этих величин и находим минимум из всех способов.

Программа

```
var
  a, b, c : array [1..100] of integer;
  n, k, i, j, s, min, l, t : integer;
function max(a,b:integer):integer;
begin
  if a>b then max:=a else max:=b
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for i:=1 to n do read(a[i]);
  read(k);
  s:=a[1]; b[1]:=a[1];
  for i:=2 to n do
  begin s:=s+a[i]; b[i]:=s end;
  for j:=2 to k do
  begin
    for i:=j to n do
    begin
      min:=maxint; s:=0;
      for l:=1 to i-j+1 do
      begin
        s:=s+a[i-l+1];
        t:=max(s,b[i-l]);
        if t<min then min:=t
      end;
      c[i]:=min
    end;
    b:=c
  end;
  write(b[n])
end.
```

Муниципальный этап 2009-2010 уч. года, 7-9 классы, 1-й тур

25. Напёрстки

(Время: 1 сек. Память: 16 Мб)

Шулер показывает следующий трюк. Он имеет три одинаковых наперстка. Под первый (левый) он кладет маленький шарик. Затем он очень быстро выполняет ряд перемещений наперстков, каждое из которых – это одно из трех перемещений - A, B, C:

- A - обменять местами левый и центральный наперстки,
- B - обменять местами правый и центральный наперстки,
- C - обменять местами левый и правый наперстки.

Необходимо определить, под каким из наперстков окажется шарик после всех перемещений.

Входные данные

В единственной строке входного файла input.txt записана строка длиной не более 50 символов из множества {A, B, C} – последовательность перемещений.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести номер наперстка, под которым окажется шарик после перемещений.

Пример

№	input.txt	output.txt
1	CBABCACCC	1

Разбор

Введём три переменные x, y, z, которые будут обозначать факт нахождения шарика (значение переменной равно единице) в левом, среднем и правом напёрстках. Вначале x=1, y=z=0. Далее обрабатываем заданные перемещения напёрстков, меняя значения соответствующих переменных.

Программа

```
var x, y, z, u, i : integer; s : string;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s); x:=1; y:=0; z:=0;
  for i:=1 to length(s) do
    case s[i] of
      'A' : begin u:=x; x:=y; y:=u end;
      'B' : begin u:=y; y:=z; z:=u end;
      'C' : begin u:=x; x:=z; z:=u end;
```

```
    end;
  if x=1 then write(1); if y=1 then write(2); if z=1 then
write(3);close(output)
end.
```

26. Игра со спичками

(Время: 1 сек. Память: 16 Мб)

Двое играют в следующую игру. Из кучки спичек за один ход игрок вытягивает либо 1, либо 2, либо 1000 спичек. Выигрывает тот, кто забирает последнюю спичку. Кто выигрывает при правильной игре?

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число — N ($1 \le N \le 10000$) начальное количество спичек в кучке.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести 1, если выигрывает первый игрок (тот, кто ходит первым), или 2, если выигрывает второй игрок.

Примеры

№	input.txt	output.txt
1	2	1
2	3	2

Разбор

Если бы можно было брать только одну или две спички, то такая игра называется игрой Баше. Для неё существует следующая выигрышная стратегия: надо брать столько спичек, чтобы осталось количество, кратное трём. И тогда, если количество спичек вначале было кратно трём, то выиграет второй игрок, иначе – первый. Проанализируем нашу игру. Если спичек меньше 1000, то надо придерживаться выигрышной стратегии игры Баше. Если спичек 1000, то выиграет первый игрок, сразу забрав их все. Но он может взять и одну спичку, тогда останется 999 спичек и он тоже выиграет. Если спичек 1001, то первый игрок, взяв две спички, сводит задачу к своему выигрышу. Если спичек 1002, то все варианты взятия спичек первым игроком приводят к его проигрышу. Дальнейший анализ показывает, что если один из игроков берёт 1000 спичек, то другой игрок, взяв одну или две спички, сводит игру к выигрышной позиции. Таким образом, в нашей игре оказывается такая же выигрышная стратегия, как и в игре Баше.

Программа

```
var n : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); if n mod 3=0 then write(2) else write(1);
close(output)
end.
```

27. Шашки

(Время: 1 сек. Память: 16 Мб)

На доске стоит белая шашка. Требуется определить, может ли она попасть в заданную клетку, делая ходы по правилам (не превращаясь в дамку).

Входные данные

В единственной строке входного файла input.txt записаны: клетка, где стоит шашка, в шахматной нотации, а затем, через пробел, клетка, куда шашка должна попасть. Начальная и конечная клетки не совпадают.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести слово YES (заглавными буквами), если шашка может попасть из начальной клетки в конечную, и NO в противном случае.

Примеры

№	input.txt	output.txt	
1	a1 b2	YES	Для выполнения указанного перемещения шашка должна сделать один ход вперед и вправо
2	b2 a1	NO	Назад шашка ходить не может
3	a1 h7	NO	a1 и h7 - клетки разного цвета
4	a1 h8	YES	7 ходов вправо вверх

Пояснение

Доска имеет размер 8x8, вертикали нумеруются маленькими латинскими буквами от a до h, горизонтали - числами от 1 до 8. Белая шашка ходит по чёрным полям по диагонали вверх.

Разбор

Переведём координаты из шахматной нотации в координатную. Далее проверяем одного ли цвета заданные поля (у полей белого цвета сумма координат является чётным числом). Если да, то проверяем,

что второе поле расположено выше, а также, что оно расположено в допустимых границах.

Программа

```
var s : string; x, y, u, v : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  x:=ord(s[1])-ord('a')+1; y:=ord(s[2])-48;
  u:=ord(s[4])-ord('a')+1; v:=ord(s[5])-48;
  if ((x+y)mod 2=0)and((u+v)mod 2=0) then
    if (v>y)and(x+y-v<=u)and(u<=x+v-y) then write('YES')
      else write('NO')
    else write('NO');
  close(output)
end.
```

28. Сумма n-значных чисел

(Время: 1 сек. Память: 16 Мб)

По заданному n найти сумму всех n-значных чисел.

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число n ($1 \le n \le 100$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести все цифры суммы всех n-значных чисел.

Примеры

№	input.txt	output.txt
1	1	45
2	2	4905

Разбор

Всего существует 900…0 (n-1 нуль) n-значных чисел от 100…0 (n-1 нуль) до 999…9 (n девяток). Если их все сложить, то получится число (100…0+999…9)*900…0/2. Проведём вычисления в указанном порядке, как учат в школе столбиком. Число в скобках будет равно 1099…9 (n-1 девятка). Умножим его на 900…0 (n-1 ноль) получим число 9899…9100..0, в котором по n-1 девятке (не считая первую) и нулю. Также школьным правилом деления столбиком поделим полученное число на два. Имеем число 49499…95500…0, в котором подряд записаны n-3 девятки и n-2 нуля. Отдельно рассматриваем одно-

и двузначные числа, так как они не подходят под полученную схему вычисления.

Программа

```
var n, i : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  if n=1 then write(45) else
  if n=2 then write(4905) else begin
    write(494); for i:=1 to n-3 do write(9);
    write(55); for i:=1 to n-2 do write(0)
  end; close(output)
end.
```

Муниципальный этап 2009-2010 уч. года, 7-9 классы, 2-й тур

29. Детали

(Время: 1 сек. Память: 16 Мб)

На клеточном поле N•M расположены две жёсткие детали. Деталь A накрывает в каждой строке несколько (не ноль) первых клеток, деталь B — несколько (не ноль) последних; каждая клетка либо полностью накрыта одной из деталей, либо нет.

A	A	.	B	B	B
A	.	.	.	.	B
A	A	A	.	.	B
A	.	.	B	B	B

Деталь B начинают двигать влево, не поворачивая, пока она не упрётся в A хотя бы одной клеткой. Определите, на сколько клеток будет сдвинута деталь B.

Входные данные

В первой строке входного файла input.txt записано два числа N и M —число строк и столбцов соответственно ($1 \le N, M \le 100$). Далее следуют N строк, задающих расположение деталей. В каждой находится ровно M символов "A" (клетка, накрытая деталью A), "B" (накрытая деталью B) или "." (свободная клетка).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число — ответ на задачу.

Пример

№	input.txt	output.txt
1	4 6 AA.BBB A....B AAA..B A..BBB	1

Разбор

Подсчитаем, сколько пустых клеток находится между двумя деталями в каждой строке. Из всех строк выберем ту, в которой это количество минимально (или одну из них, если таковых несколько). Покажем, что число пустот в этой строке и будет ответом на задачу.

Если мы сдвинули деталь B на одну клетку влево, то количество пустот в каждой строке уменьшится на единицу. Деталь упрётся лишь в том случае, когда в какой-то строке пустоты вообще исчезнут, а одной из первых таких строк будет именно выбранная. Осталось заметить, что, так как за каждый сдвиг число пустот в выбранной строке уменьшалось на единицу и в конце оно стало равным нулю, то количество сдвигов и есть начальное количество пустот в строке.

Программа

```
var
  y, x, i, j, p, min : integer; c : char;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(y,x); min:=101;
  for i:=1 to y do begin
    p:=0;
    for j:=1 to x do begin
      read(c); if c='.' then inc(p);
    end;
    if p<min then min:=p; readln;
  end;
  writeln(min); close(output)
end.
```

30. Числа без одинаковых цифр

(Время: 1 сек. Память: 16 Мб)

Антон записал ряд натуральных чисел в порядке возрастания: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 и т.д. Затем вычеркнул из него все числа, в которых имеется хотя бы

две одинаковых цифры, и получил последовательность: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 23 и т.д.

Вам необходимо по заданному N найти N-ое по счету число в получившейся последовательности.

Входные данные

В единственной строке входного файла input.txt записано натуральное число N ($1 \le N \le 10000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести N-ое по счету число без одинаковых цифр.

Пример

№	input.txt	output.txt
1	100	123

Разбор

Начиная с единицы, просматриваем числа, для каждого находим количества встречающихся в нём цифр и пропускаем те, в которых хотя бы одна цифра встречается более одного раза.

Программа

```
var   n, i, j, k, l, m : integer;
  a : array [0..9] of integer; t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); j:=0;
  for i:=1 to n do
    repeat
      j:=j+1; k:=j; for l:=0 to 9 do a[l]:=0;
      while k>0 do begin
        m:=k mod 10; a[m]:=a[m]+1; k:=k div 10
      end;
      t:=true; for l:=0 to 9 do t:=t and (a[l]<2)
    until t;
  write(j); close(output)
end.
```

31. Газон

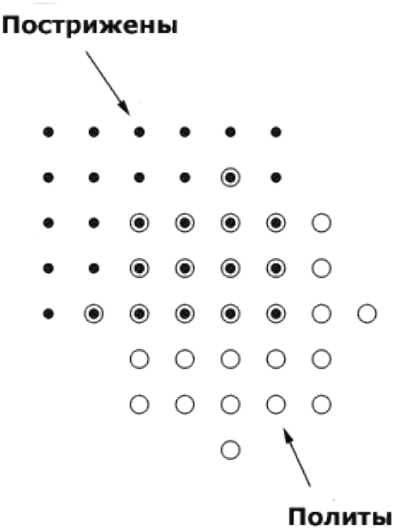
(Время: 1 сек. Память: 16 Мб)

Фермер Иван с юности следит за своим газоном. Газон можно считать плоскостью, на которой в каждой точке с целыми координатами растет один пучок травы.

В одно из воскресений Иван воспользовался газонокосилкой и постриг некоторый прямоугольный участок газона. Стороны этого

участка параллельны осям координат, а две противоположные вершины расположены в точках (x1, y1) и (x2, y2). Следует отметить, что пучки травы, находящиеся на границе этого прямоугольника, также были пострижены.

Довольный результатом Иван купил и установил на газоне дождевальную установку. Она была размещена в точке с координатами (x3, y3) и имела радиус действия струи r. Таким образом, установка начала поливать все пучки, расстояние от которых до точки (x3, y3) не превышало r.



Все было хорошо, но Ивана заинтересовал следующий вопрос: сколько пучков травы оказалось и пострижено, и полито в это воскресенье?

Требуется написать программу, которая позволит дать ответ на вопрос Ивана.

Входные данные

Первая строка входного файла input.txt содержит четыре целых числа x1, y1, x2, y2 ($-100\,000 \le x1 < x2 \le 100\,000$; $-100\,000 \le y1 < y2 \le 100\,000$). Во второй строке записаны три целых числа x3, y3, r ($-100\,000 \le x3, y3 \le 100\,000$; $1 \le r \le 100\,000$)

Выходные данные

В выходной файл output.txt необходимо вывести одно целое число – число пучков травы, которые были и пострижены, и политы.

Пример

№	input.txt	output.txt
1	0 0 5 4 4 0 3	14

Разбор

Одно из возможных решений данной задачи заключается в следующем. Для каждой прямой, содержащей все пучки травы с равной координатой по оси абсцисс (аналогично можно рассматривать и ось ординат), найдем ее точки пересечения с окружностью, отображающей область действия поливальной установки, если, конечно, такие точки существуют. Получив данные точки, не сложно за время O(1) посчитать количество пучков травы на данной прямой одновременно и политых, и постриженных. Это делается с помощью операции вычисления целой части числа. Далее, просуммировав найденные числа для всех прямых, можно найти ответ на поставленную задачу.

Не сложно показать, что время работы рассматриваемого решения будет O(r).

Программа

```
type abc=int64;
var x1,y1,x2,y2,x3,y3,x : longint; k,r,y : abc;
function kol(x,z1,z2:longint):longint;
  var min, max, k : longint;
begin
  if (x<x1)or(x>x2)or(z1>y2)or(z2<y1) then k:=0 else be-
gin
    min:=y1; if z1>y1 then min:=z1; max:=y2; if z2<y2
then max:=z2;
    k:=max-min+1
  end;
  kol:=k
end;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(x1,y1,x2,y2);
  read(x3,y3,r); k:=kol(x3,y3-r,y3+r); y:=r;
  for x:=1 to r-1 do begin
    while sqr(x)+sqr(y)>sqr(r) do y:=y-1;
    k:=k+kol(x3+x,y3-y,y3+y)+kol(x3-x,y3-y,y3+y)
  end;
  k:=k+kol(x3+r,y3,y3)+kol(x3-r,y3,y3);
  write(k); close(output)
end.
```

32. Выбор приборов

(Время: 1 сек. Память: 16 Мб)

Для проведения эксперимента надо выбрать из N имеющихся приборов только три. Для этого выполняют следующую операцию - если в группе приборов больше трех, то их нумеруют и выбирают одну из групп: с четными или нечетными номерами. Операцию повторяют до тех пор, пока в группе не останется три или менее приборов. Если их остается ровно три, то они и берутся для эксперимента.

Требуется написать программу, которая подсчитает количество способов такого выбора приборов.

Входные данные

В единственной строке входного файла input.txt записано число N ($1 \le N \le 2147483647$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - найденное количество способов выбора приборов.

Примеры

№	input.txt	output.txt
1	3	1
2	6	2

Разбор

Обозначим через f(n) – количество способов выбора приборов. Легко найти, что f(1)=f(2)=0, f(3)=1. Также легко получаем, что f(n)=f(n div 2)+f(n- n div 2) или при чётном n f(2k)=2f(k), а при нечётном n f(2k+1)=f(k)+f(k+1). Это позволяет написать приведённую ниже рекурсивную программу. Но полученные формулы позволяют решить задачу и динамически. Таким образом, пара соседних значений f(n) при n=2k и n=2k+1 выразились через пару соседних значений k и k+1. Рассмотрим случай чётного k=2m. Тогда f(n)=f(2k)=2f(k)=4f(m), а f(n)=f(2k+1)=f(k)+f(k+1)=2f(m)+f(m)+f(m+1)=3f(m)+f(m+1). Аналогично рассматривается случай и нечётного k. Таким образом, значения искомой функции пересчитывается через соседние вдвое меньшие значения, которые задаются коэффициентами i и j. Вначале эти коэффициенты определяются из соотношения f(n)=i*f(n)+j*f(n+1), т.е. равны i=1, j=0. а далее пересчитывается в зависимости от чётности либо i=2i+j, либо j=i+2j. Также, в зависимости от чётности последнего значения n выводим i или j.

Программа

```
var n : longint;
function K(n:longint):longint;
begin
  if n<3 then K:=0 else if n=3 then k:=1 else
  if odd(n) then K:=K(n div 2)+K(n div 2+1) else K:=2*K(n
div 2)
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); write(K(n))
end.

var n, i, j : longint;
begin
  assign(input,'input.txt'); reset(input);
```

```
  assign(output,'output.txt'); rewrite(output);
  read(n); i:=1; j:=0;
  while n>3 do begin
    if odd(n) then j:=i+2*j else i:=2*i+j; n:=n div 2
  end;
  if n<2 then write(0) else if n=2 then write(j) else
write (i)
end.
```

Муниципальный этап 2009-2010 уч. года, 10-11 классы, 1-й тур

33. Количество участников олимпиады

(Время: 1 сек. Память: 16 Мб)

Как известно, на вопрос о том, сколько у него учеников, древнегреческий учёный Пифагор отвечал так: "Половина моих учеников изучает математику, четвертая часть изучает природу, седьмая часть проводит время в молчаливом размышлении, остальную часть составляют 3 девы".

Секретарь олимпиады на вопрос: "Сколько участников на олимпиаде по информатике?", отвечал подобно Пифагору: "K-тая часть участников начала решать первую задачу, M-тая часть – вторую, а N-тая – третью. В то же время D участников решают проблему: "С чего начать?". Ваша задача определить количество участников олимпиады или вывести -1, если секретарь ошибся.

Входные данные

В единственной строке входного файла input.txt записаны числа K, N, M, D ($1 \le K, N, M, D \le 100$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – количество участников олимпиады или вывести -1, если секретарь ошибся.

Примеры

№	input.txt	output.txt
1	2 4 7 3	28
2	2 2 2 3	-1

Разбор

Обозначим через X общее количество участников олимпиады. Тогда X/K+X/N+X/M+D=X, из которого получаем, что X=K*N*M*D/(K*N*M-K*N-K*M-N*M). Так как решение должно быть натуральным числом, то это проверим условия неотрицательно-

сти знаменателя и делимости числителя на знаменатель в полученной дроби.

Программа

```
var
  k,m,n,d,x,y : int64;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k,m,n,d); x:=k*m*n-k*m-k*n-m*n; y:=k*n*m*d;
  if (x<=0) or (y mod x<>0) then writeln(-1) else wri-
teln(y div x);
  close(output)
end.
```

34. Садовник-художник

(Время: 1 сек. Память: 16 Мб)

Садовник посадил N деревьев в один ряд. После посадки деревьев садовнику нужно их покрасить. В его распоряжении есть краска трех цветов: белая, синяя и оранжевая. Сколько способов покраски деревьев есть у него, если никакие два соседних дерева нельзя красить в одинаковый цвет?

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число - количество деревьев N ($1 \le N \le 50$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - количество способов покраски.

Пример

№	input.txt	output.txt
1	3	12

Разбор

Первое дерево садовник может покрасить в любой из трёх цветов, а каждое из следующих только в два цвета. Всего получается $3*2^{N-1}$ вариантов. При указанных в задаче ограничениях количество вариантов не помещаются ни в тип integer, ни longint, но помещается в int64.

Программа

```
var
  k : int64; n, i : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
```

```
  read(n); k:=3; for i:=1 to n-1 do k:=k*2;
  writeln(k); close(output)
end.
```

35. Абракадабра

(Время: 1 сек. Память: 16 Мб)

Последовательность из латинских букв строится следующим образом. Вначале она пуста. На каждом последующем шаге последовательность удваивается, после чего к ней слева дописывается очередная буква латинского алфавита (a, b, c, …). Ниже приведены первые шаги построения последовательности:

Шаг 1. a

Шаг 2. baa

Шаг 3. cbaabaa

Шаг 4. dcbaabaacbaabaa

……

Требуется написать программу, которая по заданному числу N находит символ, который стоит на N-ом месте в последовательности, получившейся после 26-го шага.

Входные данные

В единственной строке входного файла input.txt записано число N ($1 \le N < 2^{26}$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести символ, стоящий в N-й позиции получившейся последовательности.

Пример

№	input.txt	output.txt
1	4	w

Разбор

На первом шаге имеем строку длины $1=2^1-1$, на втором – $3=2^2-1$, …, на 26-м - $2^{26}-1$. При этом на первом шаге в строке появляется буква a, на втором – b, …, на 26-м – z. Рассмотрим строку, которая получилась на 26 шаге. В этой строке на первом месте стоит буква z, а далее два раза записана строка с предыдущего шага. Поэтому, если N=1, то значит надо вывести в ответ букву z, а иначе надо найти место этого N-го символа в строке на предыдущем шаге. Такой возврат к предыдущему шагу продолжается до тех пор, пока N не станет равным единице.

Программа

```
var
  k : integer; n, p : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); p:=2; for k:=2 to 25 do p:=p*2; k:=0;
  while n>1 do begin
    k:=k+1; if n>p then n:=n-p else n:=n-1; p:=p div 2
  end;
  write(chr(ord('z')-k)); close(output)
end.
```

36. Змей Горыныч

(Время: 1 сек. Память: 16 Мб)

В некотором царстве жил Змей Горыныч. У него было N голов и M хвостов. Иван-царевич решил уничтожить губителя человеческих душ, для чего ему его кума Баба Яга подарила волшебный меч, так как только им можно убить Змея Горыныча. Если отрубить одну голову, то на её месте вырастает новая, если отрубить хвост, то вместо него вырастет 2 хвоста. Если отрубить два хвоста, то вырастает 1 голова, и только когда отрубить 2 головы, то не вырастет ничего. Змей Горыныч гибнет только в том случае, когда ему отрубят все головы и все хвосты. Определить минимальное количество ударов мечом, нужное для уничтожения Змея Горыныча.

Входные данные

В единственной строке входного файла input.txt записаны через пробел два числа N, M ($0 \le N, M \le 1000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – минимальное количество ударов мечом, или -1, если уничтожить Змея Горыныча невозможно.

Пример

№	input.txt	output.txt
1	3 3	9

Разбор

Пусть позиция (x, y) на целочисленной решётке в первом квадранте координатной плоскости обозначает наличие у дракона x голов и y хвостов. Согласно условиям задачи из позиции (x, y) можно попасть в одну из позиций: (x, y+1), (x+1, y-2), (x-2, y). Такой переход от содержательной постановки задачи к геометрической позволяет сразу

определить решение задачи. Например, если у Змея Горыныча нет хвостов и нечётное количество голов, то Ивану-царевичу не удастся его уничтожить. Во всех же других случаях это возможно и минимальное количество ударов мечом легко находится из указанной геометрической трактовки задачи.

Программа

```
var
  n, m, k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,m); k:=0;
  if (n mod 2=1)and(m=0) then k:=-1 else
  if n mod 2=0 then begin
    while m mod 4<>0 do begin k:=k+1; m:=m+1 end;
    k:=k+n div 2+3*(m div 4) end
  else begin
    while m mod 4<>2 do begin k:=k+1; m:=m+1 end;
    k:=k+(n+1)div 2+3*((m+2)div 4)-2 end;
  write(k); close(output)
end.
```

Муниципальный этап 2009-2010 уч. года, 10-11 классы, 2-й тур

37. Палиндромное время

(Время: 1 сек. Память: 16 Мб)

Пекарь считает, что для получения рождественского пирога идеальной симметричной формы его нужно вынимать из духовки в тот момент, когда часы показывают «палиндромное» время, которое читается одинаково слева-направо и справа-налево.

Напишите программу, которая определяет по времени установки пирога в духовку время, когда будет подходящее время для его извлечения.

Входные данные

В единственной строке входного файла input.txt записано время установки пирога в духовку в формате HH:MM ($00 \le HH \le 23$, $00 \le MM \le 59$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести ближайшее «палиндромное» время в формате HH:MM.

Примеры

№	input.txt	output.txt
1	00:00	01:10
2	12:34	13:31
3	23:59	00:00

Разбор

Введя время как строку, находим часы и минуты. В цикле с постусловием добавляем по одной минуте (не забывая пересчитывать минуты и часы при возможном переходе через час и через сутки) пока не получим «палиндромное» время.

Программа

```
var s : string; c, m : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  c:=(ord(s[1])-48)*10+ord(s[2])-48; m:=(ord(s[4])-
48)*10+ord(s[5])-48;
  repeat
    m:=m+1; if m=60 then begin m:=0; c:=c+1; if c=24 then
c:=0 end;
  until (c div 10=m mod 10)and(c mod 10=m div 10);
  if c<10 then write(0); write(c,':');
  if m<10 then write(0); write(m); close(output)
end.
```

38. Просто простые числа

(Время: 1 сек. Память: 16 Мб)

Дано натуральное число N. Представить его в виде суммы простых натуральных чисел так, чтобы произведение этих слагаемых было максимально.

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число N (1 < N < 2000000000).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести простые числа по возрастанию с указанием их количества при разложении, т.е.: <число> <количество>.

Примеры

№	input.txt	output.txt
1	5	2 1 3 1
2	30	3 10

Разбор

Проанализируем небольшие значения N. Для N меньших 6 всё очевидно. Для N=6 имеем два варианта: N=2+2+2 с произведением 8 и N=3+3 с произведением 9. Выбираем второй вариант. При N=7 также возможны два варианта: N=2+5 с произведением 10 и N=2+2+3 с произведением 12. Продолжим рассматривать варианты. При N=8 имеем: N=2+2+2+2 с произведением 16, N=2+3+3 с произведением 18, вариант N=3+5 даёт произведение 15. Таким образом, это рассмотрение наводит на мысль, что в разложении должно быть как можно больше троек. Строго это можно доказать используя неравенство из математики. Для решения задачи окончательно имеем следующее правило. Если число N нацело делится на три, то надо взять N/3 троек. Если число N при делении на три имеет остаток один, то надо взять две двойки, а остальные тройки. Если при делении N на три получается в остатке два, то надо взять одну двойку и остальные тройки.

Программа

```
var n, k, l : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); k:=n div 3; l:=0;
  if n mod 3=1 then begin k:=k-1; l:=2 end;
  if n mod 3=2 then l:=1;
  if l>0 then write('2 ',l,' '); if k>0 then write('3
',k); close(output)
end.
```

39. Спички

(Время: 1 сек. Память: 16 Мб)

Какое минимальное количество спичек необходимо для того, чтобы выложить на плоскости N квадратов со стороной в одну спичку? Спички нельзя ломать и класть друг на друга. Вершинами квадратов должны быть точки, где сходятся концы спичек, а сторонами – сами спички.

Напишите программу, которая по количеству квадратов N, которые необходимо составить, находит минимальное необходимое для этого количество спичек.

Входные данные

В единственной строке входного файла input.txt записано одно целое число N ($1 \le N \le 10^9$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число – минимальное количество спичек, требуемых для составления заданного количества квадратов.

Пример

№	input.txt	output.txt
1	4	12

Разбор

Найдём наибольшее число a, что $a^2 \le N$. Для построения квадрата со стороной a потребуется $2*(a^2+a)$ спичек. Если построены ещё не все квадраты, то достраиваем их вдоль стороны квадрата. При этом для построения первого квадрата требуется три спички, а каждого следующего ещё по две спички. Если же опять не построены все квадраты, то достраиваем их вдоль большей стороны построенного прямоугольника размером a*(a+1). При этом также потребуется три спички для первого квадрата и по две для каждого следующего.

Программа

```
var n, k, a, l : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); a:=1; while sqr(a+1)<n do a:=a+1;
  k:=2*(sqr(a)+a); l:=n-sqr(a);
  if l>0 then k:=k+2*l+1; if l>a then k:=k+1;
  write(k); close(output)
end.
```

40. Числа

(Время: 1 сек. Память: 16 Мб)

Дана последовательность чисел a_1, a_2, …, a_N. За одну операцию разрешается удалить любое (кроме крайних) число, заплатив за это штраф, равный произведению этого числа на сумму соседних. Требуется удалить все числа, кроме крайних, с минимальным суммарным штрафом.

Например:

- Начальная последовательность: **1 50 51 50 1.**
- Удаляем четвёртое число, штраф 50(51+1)=2600, получаем **1 50 51 1.**
- Удаляем третье число, штраф 51(50+1)=2601, получаем **1 50 1.**
- Удаляем второе число, штраф 50(1+1)=100.
- Итого штраф **5301.**

Входные данные

В первой строке входного файла input.txt записано одно число N ($1 \le N \le 100$) - количество чисел в последовательности.

Во второй строке находятся N целых чисел a_1, a_2, …, a_N; никакое из чисел не превосходит по модулю 100.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - минимальный суммарный штраф.

Пример

№	input.txt	output.txt
1	5 1 50 51 50 1	5301

Разбор

Задача решается методом динамического программирования.

Заведём квадратную матрицу M размера N*N. В ячейке $M_{i,j}$ будем хранить минимальный штраф за удаление всех чисел с i-го по j-е не включительно (элементы матрицы, у которых $i \ge j$, нам не будут нужны). Подсчитывать эти штрафы будем следующим образом: поскольку среди чисел между i-м и j-м какое-то (k-е) мы должны удалить последним, то переберём все k от i до j не включительно и посчитаем наименьший штраф за удаление всех чисел между i и j при условии, что k-е удаляется последним. Этот штраф равен сумме наименьших штрафов за удаление всех чисел с i по k ($M_{i,k}$), с k по j ($M_{k,j}$) и штрафа за последнее удаление - $(a_i+a_j)a_k$. Тогда очевидно, что $M_{i,j} = \min_{i \le k \le j}\left(M_{i,k} + M_{k,j} + a_k\left(a_i + a_j\right)\right)$.

Несложно посчитать элементы матрицы с индексами, отличающимися на 1: так как удалять между соседними числами нечего, то $M_{i,i+1} = 0$.

Осталось только вычислить все элементы матрицы (например, в порядке увеличения разности между индексами) и посмотреть на число $M_{1,N}$, которое и будет ответом на задачу.

Кроме того, надо аккуратно обработать случай N=1. Вышеприведённый алгоритм для этого случая не подходит, но очевидно, что, так как удалять при N=1 нечего, то ответ здесь - 0.

Программа

```
const inf=100*100*200*10;
var n, i, l, k : integer;
    a : array[1..100] of integer;
    M : array[1..100,1..100] of longint;
    min : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  for i:=1 to n do
    read(a[i]);
  for i:=1 to n-1 do
    M[i,i+1]:=0;
  for l:=3 to n do
    for i:=1 to n-l+1 do begin
      min:=inf;
      for k:=i+1 to i+l-2 do
        if min>M[i,k]+M[k,i+l-1]+
            a[k]*(a[i]+a[i+l-1]) then
          min:=M[i,k]+M[k,i+l-1]+
            a[k]*(a[i]+a[i+l-1]);
      M[i,i+l-1]:=min;
    end;
  if n<>1 then
    writeln(M[1,n])
  else writeln(0);
  close(output);
end.
```

Муниципальный этап 2010-2011 уч. года, 7-8 классы

41. Будильник

(Время: 1 сек. Память: 16 Мб)

Известный исследователь Чарльз Ф. Мантц, устав от долгого путешествия через джунгли, лег спать в 10 часов вечера, но предварительно он завел будильник на 12 часов следующего дня. Но проспать 14 часов ему не удалось – будильник зазвонил через 2 часа. Исследователь забыл, что на будильнике, имеющем 12-тичасовой циферблат, можно задать время до звонка только менее 12 часов.

Напишите программу, которая определяет, сколько часов успеет проспать исследователь, прежде чем будильник его разбудит.

Входные данные

В единственной строке входного файла input.txt записаны два целых числа S и T ($1 \le S, T \le 12$), разделенные одним пробелом - час, когда исследователь лег спать, и час, на который он установил будильник.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число – через сколько часов зазвонит будильник.

Пример

№	input.txt	output.txt
1	`10 12`	`2`

Разбор

В зависимости от значений введённых S и T вывести либо T-S, либо T+12-S.

Программа

```
var s,t:integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(s,t);
  if s<=t then writeln(t-s) else writeln(t+12-s);
  close(output);
end.
```

42. Офис

(Время: 1 сек. Память: 16 Мб)

Летом Вася очень любил смотреть в окно. Напротив его дома расположился офис некоторой строительной фирмы. В течение всего месяца Вася наблюдал за его сотрудниками. Про каждый из 31 дня месяца он знает, сколько сотрудников пришло на работу. Ему также известно, что каждый из сотрудников берет ровно по 4 выходных в месяц.

Теперь он ломает голову над загадкой – сколько всего сотрудников работает в этом офисе. Напишите программу, которая ответит Васе на этот вопрос.

Входные данные

В единственной строке входного файла input.txt записаны 31 целое неотрицательное число. Эти числа описывают количество со-

трудников, пришедших в офис в соответствующие дни месяца. Гарантируется, что входные данные корректны.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести единственное число – общее количество сотрудников офиса. Гарантируется, что ответ не превышает 100.

Пример

№	input.txt	output.txt
1	10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 0 0 0	10

Примечание

В примере все числа записаны в одной строке.

Разбор

Сумма всех чисел во входном файле равна суммарному количеству посещений офиса всеми его сотрудниками в течение месяца. А это в точности 27 умножить на количество сотрудников, поскольку каждый из них посетил офис ровно 27 раз.

Программа

```
var
  s, i, d : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  s:=0;
  for i:=1 to 31 do begin read(d); s:=s+d end;
  write(s div 27);
  close(output)
end.
```

43. Строительство школы

(Время: 1 сек. Память: 16 Мб)

В деревне Интернетовка все дома расположены вдоль одной улицы по одну сторону от нее. По другую сторону от этой улицы пока ничего нет, но скоро все будет – школы, магазины, кинотеатры и т.д.

Для начала в этой деревне решили построить школу. Место для строительства школы решили выбрать так, чтобы суммарное расстояние, которое проезжают ученики от своих домов до школы, было минимально.

План деревни можно представить в виде прямой, в некоторых целочисленных точках которой находятся дома учеников. Школу также

разрешается строить только в целочисленной точке этой прямой (в том числе разрешается строить школу в точке, где расположен один из домов – ведь школа будет расположена с другой стороны улицы).

Напишите программу, которая по известным координатам домов учеников поможет определить координаты места строительства школы.

Входные данные

В первой строке входного файла input.txt сначала записано число N — количество учеников ($1 \le N \le 100000$). Во второй строке записаны в строго возрастающем порядке координаты домов учеников — целые числа, не превосходящие $2 \cdot 10^9$ по модулю.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число — координату точки, в которой лучше всего построить школу. Если ответов несколько, выведите наибольший из них.

Примеры

№	input.txt	output.txt
1	4 1 2 3 4	3
2	3 -1 0 1	0

Разбор

Интуитивно ясно, что школу надо строить напротив одного из средних домов. Этот факт можно доказать и строго математически, но на олимпиаде этого и не требуется. Пусть это будет дом с номером m, тогда суммарное расстояние равно сумме по всем домам величин $abs(x_i-x_m)$ (абсолютных величин разности координат домов). Вычисление суммарного расстояния для всех домов по этой формуле и выбор из них минимального потребует порядка N^2 действий. При $N=10^5$ потребуется 10^{10} действий, т.е. 10 миллиардов. Поэтому рассмотрим другой алгоритм.

Если школу построить напротив первого дома, то суммарное расстояние будет равно $y_1=x_2-x_1+x_3-x_1+\ldots x_N-x_1$. Если же школу строить напротив второго дома, то по сравнению с предыдущим случаем суммарное расстояние увеличится от первого дома на x_2-x_1 и уменьшится на эту же величину для всех остальных домов, которых N-1. Т.е. $y_2=y_1+x_2-x_1-(N-1)(x_2-x_1)=y_1+(2-N)(x_2-x_1)$. Зная y_i, аналогично легко находим y_{i+1}. В этом случае суммарное расстояние увеличивается для

первых i домов на величину $x_{i+1}-x_i$ и уменьшается на эту же величину для N-i домов, т.е. $y_{i+1}=y_i+(2i-N)(x_{i+1}-x_i)$. Из всех y_i осталось выбрать минимальное (если таких несколько, то самое правое их них). Для вычисления y_1 требуется порядка N действий, при вычислении y_{i+1} через y_i надо сделать два вычитания и два умножения. Таким образом, в этом алгоритме требуется порядка N действий. Также заметим, что суммарное расстояние при заданных ограничениях задачи может не поместиться в тип longint и требуется использовать тип int64.

Последняя выведенная формула подсказывает ещё более простое решение, которое представлено во второй из привёденных ниже программ.

Программа

```
var
  n, i, xmin, d : longint;
  y, min : int64;
  x : array [1..100000] of longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); y:=0;
  for i:=1 to n do begin
    read(d); x[i]:=d; y:=y+d-x[1]
  end;
  min:=y; xmin:=x[1];
  for i:=1 to n-1 do begin
    y:=y+(2*i-n)*(x[i+1]-x[i]);
    if y<=min then begin min:=y; xmin:=x[i+1] end
  end;
  write(xmin); close(output)
end.
```

```
var
  n, i, x : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); n:=n div 2+1;
  for i:=1 to n do read(x);
  write(x); close(output)
end.
```

44. Футбол

(Время: 1 сек. Память: 16 Мб)

Вместо того чтобы делать уроки, Вася смотрел футбольный матч и записывал счет, который показывался на табло, после каждого забитого гола.

Например, у него могла получиться такая запись: 1:0, 1:1, 1:2, 2:2, 2:3.

После этого он сложил все записанные числа: 1+0+1+1+1+2+2+2+2+3=15.

По сумме, получившейся у Васи, определите, сколько всего мячей было забито в матче.

Входные данные

В единственной строке входного файла input.txt записано одно целое неотрицательное число, не превосходящее 1000 – сумма, полученная Васей.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – общее количество забитых мячей.

Пример

№	input.txt	output.txt
1	15	5

Разбор

Обозначим через S_n – сумму, посчитанную Васей, если в матче забито n голов. Перед началом матча ещё на забито голов, а поэтому n=0 и S_0=0. После каждого забитого гола сумма увеличивается на n, т.е. $S_n=S_{n-1}+n$. Имеем последовательность сумм 0, 1, 3, 6, 10, 15, …. В математике эти числа называются треугольными. Таким образом, для решения задачи будем вычитать из заданного числа 1, 2, 3, …, пока число не станет равным нулю. Последнее вычитаемое и будет ответом.

Программа

```
var
  n, k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); k:=0;
  while n>0 do begin
    k:=k+1; n:=n-k
  end;
  write(k);
  close(output)
end.
```

Муниципальный этап 2010-2011 уч. года, 9-11 классы

45. Две цифры

(Время: 1 сек. Память: 16 Мб)

Сколько N-значных чисел можно составить, используя цифры 5 и 9, в которых три одинаковые цифры не стоят рядом?

Входные данные

В единственной строке входного файла input.txt записано одно число N ($1 \le N \le 30$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - количество чисел с указанным свойством.

Пример

№	input.txt	output.txt
1	3	6

Разбор

Задача решается методом динамического программирования. Обозначим через $f55_n$, $f59_n$, $f95_n$, $f99_n$ – количество n-значных чисел, заканчивающихся на 55, 59, 95, 99 соответственно. При n=2 имеем $f55_2=f59_2=f95_2=f99_2=1$. Также заметим, что $f55_n=f95_{n-1}$, $f59_n=f55_{n-1}+f95_{n-1}$, $f95_n=f59_{n-1}+f99_{n-1}$, $f99_n=f59_{n-1}$. Выписанные формулы позволяют последовательно находить введённые величины. Ответом будет величина $f55_N+f59_N+f95_N+f99_N$, а с учётом выписанных выше равенств $f55_{N-1}+2f59_{N-1}+2f95_{N-1}+f99_{N-1}$. Также надо учесть, что при N=1 искомых чисел 2, а при N=2 – 4.

Программа

```
var
  n, i : integer;
  nn, n55, n59, n95, n99 : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  if n<3 then writeln(2*n) else begin
    n55:=1; n59:=1; n95:=1; n99:=1;
    for i:=3 to n-1 do begin
      nn:=n59;   n59:=n55+n95;   n55:=n95;   n95:=nn+n99;
n99:=nn
    end;
    nn:=n55+2*n59+2*n95+n99;
    writeln(nn);
  end;
end.
```

46. Турнир

(Время: 1 сек. Память: 16 Мб)

Турнир проходит по олимпийской системе. В каждом матче участвуют два игрока. Проигравший игрок выбывает из турнира, а победитель проходит в следующий тур. Матчи продолжаются до тех пор, пока в турнире не останется один участник, который становится обладателем золотой медали. Серебро достаётся его оппоненту в финальном матче. Если количество участников больше трёх, то назначается дополнительный матч для определения бронзового победителя. В нём участвуют два игрока, выбывшие из турнира последними, не считая финалистов.

Напишите программу определения минимального количества матчей, которые необходимо сыграть участникам турнира, чтобы определить из них тех, кто получит медали.

Входные данные

В единственной строке входного файла input.txt записано одно целое число N ($0 \le N \le 2147483647$) – количество участников турнира.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - минимальное количество матчей в этом турнире.

Примеры

№	input.txt	output.txt
1	3	2
2	4	4

Разбор

Если участников нет или один, то матчи не проводятся. При двух участниках проводится один матч, а при трёх участниках – два. При большем количестве участников количество матчей равно количеству участников. Докажем это. Для этого обозначим через f(n) – количество матчей для определения только обладателя золотой медали. При чётном количестве участников в туре n=2k в следующий тур проходит k из них, при нечётном количестве n=2k+1 – k+1. При этом играется k матчей. Таким образом, имеем f(2k)=k+f(k), f(2k+1)=k+f(k+1), f(1)=0. Подстановкой в каждое из полученных соотношений легко проверить, что f(n)=n-1. Так как проводится ещё дополнительный матч для определения бронзового победителя, то общее количество матчей равно количеству участников.

Программа

```
var
  n : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n);
  if n<2 then write(0) else
    if n<4 then write(n-1) else write(n);
  close(output)
end.
```

47. Укладка плитки

(Время: 1 сек. Память: 16 Мб)

Бригаде строителей поручили уложить квадратной плиткой пол на кухне в виде шахматного узора. Но строители работали не очень слаженно, и когда весь пол уже был уложен, оказалось, что в некоторых местах плитки одинакового цвета граничат друг с другом.

По заданному замощению определите, какое минимальное число строителей могло укладывать плитку.

Входные данные

Входной файл input.txt содержит восемь строк, состоящих из восьми символов “W” и “B” – полученное замощение. Символ “W” обозначает плитку белого цвета, а символ “B” – чёрную.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число - искомое число строителей.

Пример

№	input.txt	output.txt
1	WBWBWBBW BWBBWBWB WBWWBWBW WBWWBWWB BWBBWBWB WBWBWWBW BWBWBBWB WBWBWWBW	4

Разбор

Просматривая заданное замощение, находим ещё не удалённые плитки и, найдя одну из них, увеличиваем количество строителей на один, а также удаляем её и все плитки, положенные правильно отно-

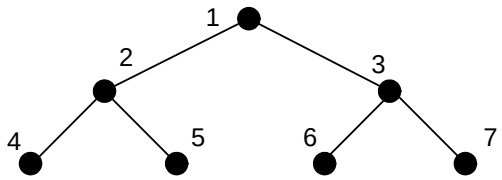
сительно неё. Для осуществления последнего действия применяем рекурсию.

Программа

```
var
  i, j, k : integer;
  a : array [0..9,0..9] of char;
procedure U(i,j:integer);
  var c : char;
begin
  c:=a[i,j]; a[i,j]:=' ';
  if (c='W')and(a[i+1,j]='B') then U(i+1,j);
  if (c='B')and(a[i+1,j]='W') then U(i+1,j);
  if (c='W')and(a[i-1,j]='B') then U(i-1,j);
  if (c='B')and(a[i-1,j]='W') then U(i-1,j);
  if (c='W')and(a[i,j+1]='B') then U(i,j+1);
  if (c='B')and(a[i,j+1]='W') then U(i,j+1);
  if (c='W')and(a[i,j-1]='B') then U(i,j-1);
  if (c='B')and(a[i,j-1]='W') then U(i,j-1);
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  for i:=1 to 8 do begin
    for j:=1 to 8 do read(a[i,j]);
    readln
  end;
  for i:=1 to 8 do begin
    a[0,i]:=' '; a[9,i]:=' ';
    a[i,0]:=' '; a[i,9]:=' '
  end;
  k:=0;
  for i:=1 to 8 do
    for j:=1 to 8 do
      if a[i,j]<>' ' then begin
        k:=k+1; U(i,j)
      end;
  write(k);
  close(output)
end.
```

48. Эволюция

(Время: 1 сек. Память: 16 Мб)



Во время исследований, посвященных появлению жизни на планете Олимпия, учеными было сделано несколько сенсационных открытий:

1. Все живые организмы планеты происходят от бактерии Bitozoria Programulis.

2. Эволюция происходила шаг за шагом (по предположению ученых – во время изменения климата на планете).

3. На каждом шаге эволюции из каждого вида образовывались ровно два подвида, а предыдущий вид исчезал.

4. Если считать появление бактерии Bitozoria Programulis первым шагом эволюции, то существующие сейчас живые организмы находятся на N-ом шаге.

Чтобы не придумывать названия во время исследований, ученые пронумеровали все виды организмов, которые когда-либо существовали на планете. Для этого они нарисовали дерево эволюции с корнем Bitozoria Programulis, которая получила номер 1. Далее нумеровали виды каждого шага эволюции слева направо. Таким образом непосредственные подвиды Bitozoria Programulis получили номера 2 и 3. Следующими были занумерованы виды третьего шага эволюции – подвиды вида 2 получили номера 4 и 5, а вида 3 – номера 6 и 7, и т.д.

Напишите программу, которая по номерам двух видов вычислит номер вида их ближайшего общего предка в дереве эволюции.

Входные данные

В первой строке входного файла input.txt записано целое число N ($1 \le N \le 60$) – количество этапов эволюции, которые произошли на планете Олимпия до текущего времени. Вторая и третья строки содержат по одному натуральному числу, которые представляют номера видов, для которых требуется найти номер их ближайшего общего предка.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно натуральное число – номер ближайшего предка для двух видов.

Примеры

№	input.txt	output.txt
1	4 15 12	3
2	18 233016 233008	14563

Разбор

Описанный в задаче способ нумерации видов позволяет легко находить по номеру подвида номер вида, из которого произошёл этот подвид. Достаточно номер подвида нацело поделить на два. Таким образом, будем перемещаться по дереву эволюции, пока не найдем совпадение номеров видов-предков для двух заданных видов.

Программа

```
var
  n, a, b : int64;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,a,b);
  while a<>b do
    if a>b then a:=a div 2 else b:=b div 2;
  write(a);
  close(output)
end.
```

Муниципальный этап 2011-2012 уч. года, 7-8 классы

49. Торт

(Время: 1 сек. Память: 16 Мб)

Никифор на день рождения собирается угостить друзей тортом. Известно, что на дне рождения может быть либо M, либо N человек, включая самого именинника. На какое минимальное количество частей ему нужно разрезать торт (не обязательно всех равных), чтобы при любом из указанных количеств собравшихся, все съели торт поровну?

Входные данные

В единственной строке входного файла input.txt записаны два натуральных числа M и N через пробел ($1 \le M, N \le 30000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести единственное число – искомое минимальное количество кусочков торта.

Пример

№	input.txt	output.txt
1	2 3	4

Пояснение

Торт нужно разрезать на части 1/3, 1/3, 1/6 и 1/6. Тогда при 2-х участниках праздника каждый съест по 1/3 + 1/6, а при 3-х каждый съест соответственно: 1/3, 1/3, 1/6 + 1/6 = 1/3.

Разбор

Мысленно представим, что у нас два торта. Один разрежем на M равных кусочков, а второй на N равных кусочков. После этого совместим торты так, чтобы у них совпал хотя бы один разрез. Но при этом могут совпасть ещё несколько разрезов, а таким образом, количество кусков уменьшится и их общее количество будет равно M + N – НОД(M, N). Для нахождения наибольшего общего делителя применим алгоритм Евклида.

Программа

```
var
  m, n, p, r : integer;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(m, n);
  r:=m+n;
  while n>0 do begin
    p:=m mod n; m:=n; n:=p
  end;
  write(r-m);
  close(output)
end.
```

50. Степенные числа

(Время: 1 сек. Память: 16 Мб)

Число *n* называется степенным, если его можно получить из некоторого числа умножением на себя хотя бы один раз. Например, 4 степенное число, так как 4=2·2, 27 тоже степенное число, так как 27=3·3·3, а 28 не является степенным числом. Определить являются ли заданные числа степенными.

Входные данные

В первой строке входного файла input.txt записано одно натуральное число n - количество исследуемых чисел ($1 \le n \le 10$). Во второй строке через пробел записаны n чисел - исследуемые числа. Каждое из них больше 1 и меньше 10^9.

Выходные данные

Выходной файл output.txt содержит n строк. В i-й строке записано “YES”, если i-е число является степенным и “NO” иначе.

Пример

№	input.txt	output.txt
1	2 27 28	YES NO

Разбор

Для определения того, что число n степенное найдём его разложение на простые множители, т.е. $n=2^{a_1}3^{a_2}\ldots p_k^{a_k}$, где $a_i > 0$ для i=1, …, k, k – количество простых делителей. Нас в этом разложении будут интересовать только числа a_1, a_2, …, a_k. Если хотя одно из них равно 1, то число не является степенным. Таким образом, из этих чисел находим минимальное min и, если оно равно 1, то число не является степенным. Для того чтобы число было степенным надо, чтобы все числа a_1, a_2, …, a_k делились на min без остатка. Проверяем это условие и печатаем соответствующее сообщение.

Программа

```
var
  n, k, i, min : integer;
  a : array [1..20] of longint;
  x, y : longint;
  t : boolean;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(n);
  while n>0 do begin n:=n-1;
    read(x); k:=0; min:=100;
    i:=0; while x mod 2=0 do begin i:=i+1; x:=x div 2
end;
    if i>0 then begin k:=k+1; a[k]:=i; if i<min then
min:=i end;
    y:=3;
    while (x>1)and(y*y<=x) do begin
      i:=0; while x mod y=0 do begin i:=i+1; x:=x div y
end;
```

```
      if i>0 then begin k:=k+1; a[k]:=i; if i<min then
min:=i end;
      y:=y+2;
    end;
    if x>1 then begin k:=k+1; a[k]:=1; min:=1 end;
    if min=1 then writeln('NO') else begin
      t:=true; for i:=1 to k do t:=t and (a[i] mod
min=0);
      if t then writeln('YES') else writeln('NO')
    end
  end;
  close(output)
end.
```

51. Колокол

(Время: 2 сек. Память: 16 Мб)

Требуется написать программу, которая в массиве из n целых чисел наименьший элемент поместит на первое место, наименьший из оставшихся – на последнее, следующий по величине – на второе место, следующий – на предпоследнее и так далее – до середины массива.

Входные данные

В первой строке входного файла input.txt записано число n (1 □*n* □30000). Во второй строке записаны через пробел элементы массива, числа по абсолютной величине не большие 32767.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести элементы полученного массива, разделенные одним пробелом.

Пример

№	input.txt	output.txt
1	5 1 2 3 4 5	1 3 5 4 2

Разбор

В переменной l будем хранить левую границу обработки массива, в r – правую, а в логической переменной t будем указывать на левую или правую границу надо перемещать минимальный элемент (t=true, если надо перемещать на левую границу и t=false иначе). Вначале l=1, r=n, t=true. Далее на отрезке от l до r находим минимальный элемент и помещаем его на левую или правую границу, не забывая при этом поменять соответствующую границу и указатель t. Очевидно, что указанные действия надо проделать n-1 раз.

Программа

```
var
  n, i, k, l, r, j, min : integer;
  a : array [1..32000] of integer;
  t : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); for i:=1 to n do read(a[i]);
  l:=1; r:=n; t:=true;
  for i:=1 to n-1 do
  begin
    min:=a[l]; k:=l;
    for j:= l+1 to r do
      if a[j]<min then begin min:=a[j]; k:=j end;
    if t then begin a[k]:=a[l]; a[l]:=min; l:=l+1 end
         else begin a[k]:=a[r]; a[r]:=min; r:=r-1 end;
    t:=not t
  end;
  for i:=1 to n-1 do write(a[i],' '); write(a[n])
end.
```

52. Ленточка

(Время: 1 сек. Память: 16 Мб)

Расположенную вертикально прямоугольную бумажную ленточку с закрепленным нижним концом стали складывать следующим образом:

- на первом шаге ее согнули пополам так, что верхняя половина легла на нижнюю либо спереди (P - сгибание) либо сзади (Z - сгибание),

- на последующих n-1 шагах выполнили аналогичное действие с получающейся на предыдущем шаге согнутой ленточкой, как с единым целым.

Затем ленточку развернули, приведя ее в исходное состояние. На ней остались сгибы - ребра от перегибов, причем некоторые из ребер оказались направленными выпуклостью к нам (K - ребра), а некоторые – от нас (O - ребра). Ребра пронумеровали сверху вниз числами от 1 до 2^n-1.

Требуется написать программу, которая по заданным строке символов из прописных букв "P" и "Z", определяющей последовательность типов сгибаний, и номерам рёбер сообщает тип этих ребер, получившийся после этой последовательности сгибаний.

Входные данные

Входной файл input.txt содержит в первой строке число n – количество сгибаний ленточки (n не более 60), во второй строке - набор n символов из прописных латинских букв "P" и "Z". Третья строка содержит в начале число k – количество рассматриваемых рёбер (не более 10), а далее их номера (числа от 1 до 2^n-1).

Выходные данные

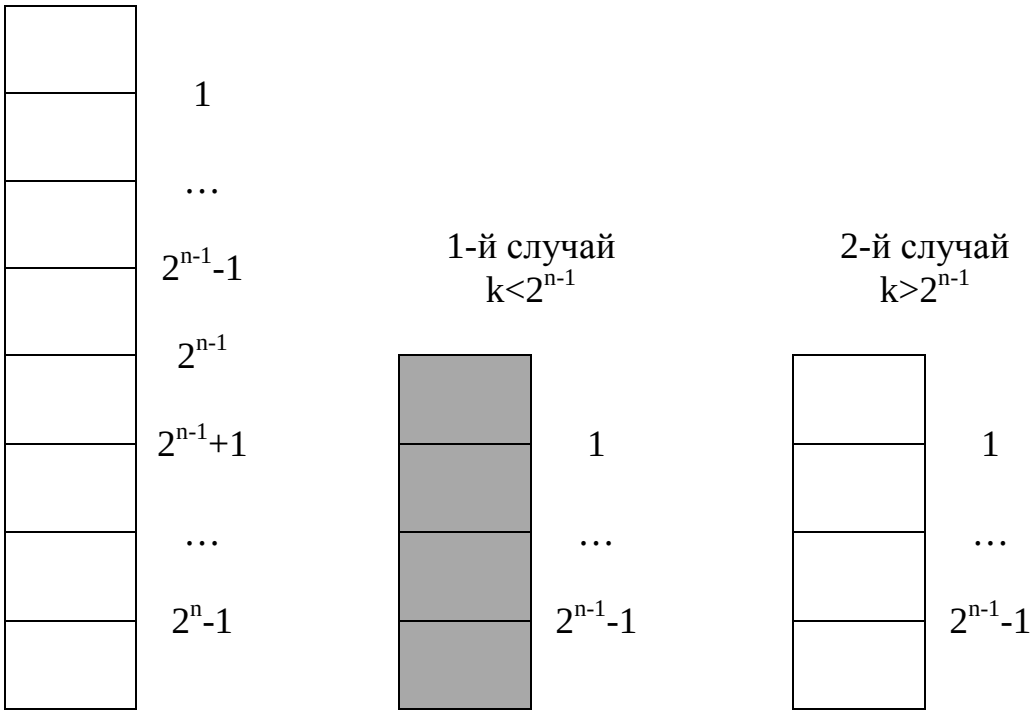
В единственную строку выходного файла output.txt нужно вывести k символов (прописные латинские буквы "K" или "O") – типы рассматриваемых ребра.

Примеры

№	input.txt	output.txt
1	2 PP 1 1	K
2	2 ZZ 2 1 2	OK

Разбор

Решим задачу методом уменьшения размера – одним из распространённых методов решения задач в математике и информатике. Для этого будем считать, что бумажная ленточка с одной стороны окрашена 1-м цветом (белым), а с другой – 2-м (тёмным). Обозначим через Z(n, k, c) решение задачи поиска типа k-го ребра при n сгибаниях в том случае, если полоска лежит к нам цветом c.



Из рисунка видно, что если $k= 2^{n-1}$, то тип сгиба можно определить по имеющейся информации о 1-м сгибании. Если $k<2^{n-1}$, то задача Z(n, k, c) имеет такое же решение как задача $Z(n-1, 2^{n-1}-k, 3-c)$. Если $k>2^{n-1}$, то задача Z(n, k, c) имеет такое же решение как задача $Z(n-1, k-2^{n-1}, c)$. Таким образом, мы или уже имеем решение задачи или свели её по 1-му параметру на единицу меньшему, по 2-му не менее вдвое меньшему и, возможно, третьему изменённому (с 1 на 2 или наоборот). Продолжим сведение задач и получим решение максимум за n таких сведений.

Программа

```
var
  k, st, t : int64;
  n, cv, i, j : integer; s : string;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(n); readln(s); read(j);
  st:=1; for i:=2 to n do st:=st*2;
  while j>0 do begin
    j:=j-1; t:=st; i:=1; cv:=1;
    read(k);
    while k<>t do begin
      if k>t then k:=k-t else begin cv:=3-cv; k:=t-k end;
      t:=t div 2; i:=i+1
    end;
    case cv of
```

```
      1 : if s[i]='P' then write('O') else write('K');
      2 : if s[i]='P' then write('K') else write('O');
    end
  end;
  close(output)
end.
```

Муниципальный этап 2011-2012 уч. года, 9-11 классы

53. Баланс скобок

(Время: 1 сек. Память: 16 Мб)

Дана последовательность, состоящая из открывающихся и закрывающихся круглых, квадратных и фигурных скобок.

Требуется написать программу, которая определит можно ли добавить в эту последовательность цифры и знаки арифметических действий так, чтобы получилось правильное скобочное выражение.

Входные данные

Входной файл input.txt состоит из хотя бы одной и не более 10 строк. В каждой строке записана одна последовательность скобок. Длина последовательности не более 255.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести символы 0 или 1. Их общее количество равно количеству введённых строк. Для каждой строки выводится 0, если из неё может получиться правильное скобочное выражение, и 1, иначе.

Пример

№	input.txt	output.txt
1	([{}]) ([{	01

Разбор

Будем просматривать строку слева направо по символам. Если очередной символ является открывающейся скобкой, то добавляем её в изначально пустой символьный массив. Если же это закрывающаяся скобка, то перед ней должна стоять аналогичная открывающаяся скобка, а предыдущую открывающуюся скобку мы занести в массив последней. Таким образом, если эти скобки соответствуют, то отбрасываем их, в частности из массива удаляем последнюю добавленную скобку. Иначе заканчиваем просмотр и выдаём сообщение о неправильности скобочного выражения. Такое же сообщение выдаём, если по окончании просмотра в массиве ещё будут скобки. Кстати, такая

структура данных, работа с которой осуществляется с одного конца, называется в программировании стеком.

Программа

```
var
  s : string;
  t : array [1..255] of char;
  l, i, k : integer;
  v : boolean;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  while not eof(input) do begin
    readln(s);
    l:=length(s); i:=0; k:=0; v:=true;
    while v and (i<l) do
    begin
      i:=i+1;
      case s[i] of
        '(', '[', '{' : begin k:=k+1; t[k]:=s[i] end;
        ')' : if k=0 then v:=false else
                if t[k]='(' then k:=k-1 else v:=false;
        ']' : if k=0 then v:=false else
                if t[k]='[' then k:=k-1 else v:=false;
        '}' : if k=0 then v:=false else
                if t[k]='{' then k:=k-1 else v:=false;
      end
    end;
    if v and (k=0) then write(0) else write(1)
  end; close(output)
end.
```

54. Три грибника

(Время: 1 сек. Память: 16 Мб)

Три грибника Петя, Вася и Коля, возвращаясь из лесу домой, решили устроить привал, а заодно и перекусить. Как это у нас принято, через некоторое время каждый начал хвастаться своими сегодняшними успехами, а потом делиться найденными грибами со своими товарищами. Изначально у каждого из них было некоторое целое количество грибов. Сначала Петя дал Васе и Коле по столько грибов, сколько у них уже было. Коля быстро понял, что так будет не по-братски, и дал Васе и Пете по столько грибов, сколько у них стало. Вася не мог отстать от сотоварищей и тоже дал каждому из друзей по столько

грибов, сколько у них к этому моменту имелось. И тут друзья с удивлением обнаружили, что у всех стало грибов поровну.

Известно, что все вместе они собрали N грибов. Сколько грибов было у каждого из них перед привалом?

Входные данные

В единственной строке входного файла input.txt записано одно натуральное число N ($N \leq 30000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести три числа через пробел – первоначальное количество грибов у Пети, Васи и Коли соответственно. Предполагается, что ответ для данного N существует.

Пример

№	input.txt	output.txt
1	120	65 20 35

Разбор

Обозначим через N общее количество собранных друзьями грибов, а тройкой чисел (x, y, z) будем обозначать количество грибов у Пети, Васи и Коли соответственно. После того как Вася поделился грибами с друзьями у них стало (N/3, N/3, N/3) грибов. Так как Вася дал Пете и Коле столько грибов, сколько у них было, то это значит, что у них было по N/6 грибов, а остальные были у Васи – (N/6, 4N/6, N/6). Тогда, аналогично, до того как поделился грибами Коля у них было (N/12, 4N/12, 7N/12) грибов. А тогда изначально у них было (13N/24, 4N/24, 7N/24) грибов. Так как решение по условию задачи всегда существует, то этим условием будет делимость количества грибов на 24 и, тогда введённое число делим нацело на 24 и полученное число умножаем на 13, 4 и 7.

Программа

```
var
  n : integer;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n);
  n:=n div 24;
  write(13*n,' ',4*n,' ',7*n);
  close(output)
end.
```

55. Бессмыслица

(Время: 1 сек. Память: 16 Мб)

Никифор утверждал, что бессмыслица, повторенная много раз, становится истиной. Для доказательства этого он применил следующую процедуру: переставил на клавиатуре своего компьютера клавиши в произвольном порядке и набрал некоторый текст. Получилась, естественно, бессмыслица. Он и эту бессмыслицу набрал на том же компьютере с той же подправленной клавиатурой. Новую бессмыслицу Никифор набрал еще раз и так далее – времени-то у него много.

Требуется написать программу, которая найдет максимальное количество шагов его процедуры, чтобы получился исходный текст.

Входные данные

В единственной строке входного файла input.txt записано одно целое число N ($1 < N < 60$) – количество клавиш на клавиатуре компьютера Никифора.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно целое число – максимальное количество шагов проделанной Никифором процедуры.

Пример

№	input.txt	output.txt
1	4	4
2	5	6

Разбор

Рассмотрим случай N=5. Если клавиши переставить сдвигом на одну, то после пяти наборов текста получил исходный текст. Если клавиши разбить на две группы по три и две клавиши и в каждой группе клавиши сдвинуть на одну, после 6 наборов получим исходный текст. Таким образом, имеем, что если $N=a_1+a_2+\ldots+a_k$, где все $a_i>0$, то количество наборов текста равно наименьшему общему кратному величин a_1, a_2, …, a_k. Рассмотрев все возможные разложения, найдем максимум из наименьших общих кратных. Это и будет ответом.

Для получения всех разложений воспользуемся алгоритмом из книги В. Липского «Комбинаторика для программистов» (стр. 54-58).

Программа

```
var
  n, d, l, sum, i : integer;
  S, R : array [1..80] of integer;
  Res, Res1 : longint;
function NOD(a,b:longint):longint;
  var c : longint;
begin
  while b>0 do
  begin c:=a mod b; a:=b; b:=c end;
  NOD:=a
end;
function NOK(a,b:longint):longint;
begin
  NOK:=a*b div NOD(a,b)
end;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n); S[1]:=n; R[1]:=1; d:=1; Res:=n;
  while S[1]>1 do
  begin
     sum:=0;
     if S[d]=1 Then begin sum:=sum+R[d]; d:=d-1 end;
     sum:=sum+S[d]; R[d]:=R[d]-1; l:=S[d]-1;
     if R[d]>0 then d:=d+1;
     S[d]:=l; R[d]:=sum div l; l:=sum mod l;
     if l<>0 then begin d:=d+1; S[d]:=l; R[d]:=1 end;
     Res1:=S[1];
     for i:=2 to d do
        Res1:=NOK(Res1,S[i]);
     if Res1>Res then Res:=Res1
  end;
  write(Res); close(output)
end.
```

56. Ленточка - 2

(Время: 1 сек. Память: 16 Мб)

Расположенную вертикально прямоугольную бумажную ленточку с закрепленным нижним концом стали складывать следующим образом:

- на первом шаге ее согнули пополам так, что верхняя половина легла на нижнюю либо спереди (P - сгибание) либо сзади (Z - сгибание),

- на последующих n-1 шагах выполнили аналогичное действие с получающейся на предыдущем шаге согнутой ленточкой, как с единым целым.

Затем ленточку развернули, приведя ее в исходное состояние. На ней остались сгибы - ребра от перегибов, причем некоторые из ребер оказались направленными выпуклостью к нам (K - ребра), а некоторые – от нас (O - ребра). Ребра пронумеровали сверху вниз числами от 1 до 2^n-1.

Требуется написать программу, которая по заданной строке символов из прописных букв "O" и "K", где нахождение на i -ом месте символа "O" или "K" определяет тип ребра на расправленной полоске, находит строку из прописных "P" и "Z", определяющих последовательность типов сгибаний, посредством которых получена ленточка с этой последовательностью ребер.

Входные данные

В первой строке входного файла input.txt записано число n – количество сгибаний (n не более 20), во второй строке – строка из 2^n-1 символов "O" или "K", определяющих типы ребер на расправленной ленточке.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести строку из n символов "P" и "Z", задающую последовательность сгибаний. Если такой последовательности сгибаний не существует, то вывести в файл "NO".

Примеры

№	input.txt	output.txt
1	2 OOK	PZ
2	2 OOO	NO

Разбор

Эта задача является обратной к задаче «Ленточка» для 7-8 классов. Для понимания предлагаемого решения советуем обратиться к разбору этой задачи на страницах 71-72, чтобы знать, как решать прямую задачу.

И так мы умеем решать прямую задачу. Покажем, как теперь решить поставленную задачу. Для этого занёсем в искомую строку из n символов символы “N”, обозначающие неопределённость соответствующего сгибания ленточки. В цикле от 1 do 2^n-1 для каждого из сги-

бов решаем прямую задачу и из типа сгиба определяем тип сгибания. Если этот тип сгибания ещё не определён, то меняем его на найденный. Если же тип сгибания уже был определён, и они не совпадают, то это значит, что последовательности сгибаний построить нельзя.

Программа

```
var
  k, k1, st, t : longint;
  n, cv, i, j : integer;
  s : string;
  c, c1 : char;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  readln(n);
  st:=1; for i:=2 to n do st:=st*2;
  s:=''; for i:=1 to n do s:=s+'N';
  for k1 :=1 to 2*st-1 do begin
    k:=k1; t:=st; i:=1; cv:=1;
    read(c);
    while k<>t do begin
      if k>t then k:=k-t else begin cv:=3-cv; k:=t-k end;
      t:=t div 2; i:=i+1
    end;
    case cv of
      1 : if c='O' then c1:='P' else c1:='Z';
      2 : if c='K' then c1:='P' else c1:='Z';
    end;
    if s[i]='N' then s[i]:=c1 else
      if s[i]<>c1 then begin write('NO'); close(output);
exit end
  end;
  writeln(s); close(output)
end.
```

Муниципальный этап 2012-2013 уч. года, 7-8 классы

57. Строки в книге

(Время - 1 сек., память - 16 Мб)

В книге на одной странице помещается k строк. Таким образом, на 1-й странице печатаются строки с 1-й по k-ю, на второй — с (k+1)-й по (2·k)-ю и т.д. Напишите программу, которая по номеру строки в тексте определяет номер страницы, на которой будет напечатана эта строка, и порядковый номер этой строки на странице.

Входные данные

В единственной строке входного файла input.txt записаны два числа: k — количество строк, которое печатается на странице, и n — номер строки ($1 \le k \le 200$, $1 \le n \le 20000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести два числа — номер страницы, на которой будет напечатана эта строка, и номер строки на странице.

Примеры

№	input.txt	output.txt
1	50 1	1 1
2	20 25	2 5
3	14 41	3 13

Разбор

Сделаем такое схематичное изображение первых двух страниц книги

Заданная нумерация строк	Нумерация, увеличенная на k-1	1-я страница	Заданная нумерация строк	Нумерация, увеличенная на k-1	2-я страница	
1	k		k+1	2k		…
2	k+1		k+2	2k+1		
3	k+2		k+3	2k+2		
…	…		…	…		
k	k+k-1		2k	2k+k-1		

Отсюда видно, что поделив нацело на k число из увеличенного на k-1 заданного номера строки, будем получать номер страницы. Аналогично можно убедиться в том, что если от номера строки убрать единицу, взять остаток от деления на k, а потом увеличить на единицу, то будем получать номер строки на соответствующей странице.

Программа на Паскале

```
var
  k, n : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k,n);
  write((n+k-1)div k,' ',(n-1)mod k+1);
  close(output)
end.
```

58. Фибоначчиева последовательность

(Время - 1 сек., память - 16 Мб)

Последовательность чисел a_1, a_2, …, a_i,… называется фибоначчиевой, если для всех $i \geq 3$ верно, что $a_i=a_{i-1}+a_{i-2}$, то есть каждый член последовательности (начиная с третьего) равен сумме двух предыдущих.

Ясно, что, задавая различные числа a_1 и a_2 , мы можем получать различные такие последовательности, и любая фибоначчиева последовательность однозначно задается двумя своими первыми членами.

Будем решать обратную задачу. Вам будет дано число n и два члена последовательности: a_n и a_{n+1}. Вам нужно написать программу, которая по их значениям найдет a_1 и a_2.

Входные данные

В единственной строке входного файла input.txt записаны: число n и значения двух членов последовательности: a_n и a_{n+1} ($1 \leq n \leq 30$, члены последовательности — целые числа, по модулю не превышающие $2 \cdot 10^9$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести два числа — значения первого и второго членов этой последовательности.

Пример

№	input.txt	output.txt
1	4 3 5	1 1

Разбор

Рассмотрим следующую стандартную схему вычисления чисел Фибоначчиевой последовательности по заданным a_1 и a_2.

a_1	a_2	$a_3= a_1+ a_2$	…	…	$a_n= a_{n-2}+ a_{n-1}$	$a_{n+1}= a_{n-1}+ a_n$
Заданы		…	…	…	…	
a	b	c=a+b				
	a=b	b=c	c=a+b			

А теперь, зная a_n и a_{n+1}, развернём вычисления в обратном порядке.

$a_1= a_3- a_2$	$a_2= a_4- a_3$		$a_{n-2}= a_n- a_{n-1}$	$a_{n-1}= a_{n+1}- a_n$	a_n	a_{n+1}
	…	…	…	…	*Заданы*	
				c=b-a	a	b
			c=b-a	c=b	b=a	

После этого уже легко написать программу вычислений.

Программа на Паскале

```
var
  n, i : integer;
  a, b, c : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,a,b);
  for i:=n-1 downto 1 do begin
    c:=b-a; b:=a; a:=c
  end;
  write(a,' ',b);
  close(output)
end.
```

59. Сжатие бинарных последовательностей

(Время - 1 сек., память - 16 Мб)

Последовательность из символов ‘0’ и ‘1’ называется бинарной. Они широко применяются в информатике и других науках. Одно из неудобств бинарных последовательностей – их трудно запоминать. Для решения этой проблемы были предложены разные способы их сжатия. Программист Слава использует следующий способ: просматривая последовательность слева направо, он заменяет ‘1’ на ‘a’, ‘01’ на ‘b’, ‘001’ на ‘c’, …, ‘00000000000000000000000001’ на ‘z’. Напишите программу, которая поможет Славе автоматизировать этот способ сжатия.

Входные данные

В единственной строке входного файла input.txt записана одна бинарная последовательность – строка из 0 и 1, длиной не более 255 символов. Гарантируется, что к ней применим указанный способ сжатия.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одну строку из латинских строчных букв от ‘a’ до ‘z’ – сжатие заданной бинарной последовательности.

Пример

№	input.txt	output.txt
1	101	ab
2	101001	abc
3	00000000000000000000000001	y

Разбор

Просматривая бинарную последовательность слева направо, считаем количество нулей до тех пор пока не встретится единица. По количеству нулей определяем получаемый символ, для этого воспользуемся стандартными функциями Паскаля ord и chr. Просмотр продолжаем, обнуляя счётчик нулей, пока не будет исчерпана строка.

Программа на Паскале

```
var
  s : string;
  i, k : integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  readln(s);
  i:=0; k:=0;
  while i<length(s) do begin
    i:=i+1;
    if s[i]='0' then k:=k+1
      else begin write(chr(ord('a')+k)); k:=0 end
  end;
  close(output)
end.
```

60. Вирусы

(Время - 2 сек., память - 16 Мб)

Для моделирования различных объектов часто применяются так называемые клеточные поля. В простейшем случае – это прямоугольные таблицы, характеризующие некоторую область, а в каждой ячейке таблицы записывается какая-либо информация об исследуемом объекте. В биологии для моделирования распространения вирусов на плоской области в каждой ячейке помечается наличие вируса, а его распространение осуществляется в соседние ячейки по вертикали и горизонтали за одну единицу времени. В начальный момент времени в исследуемую область проникли несколько вирусов. Напишите программу, которая найдёт время заражения всей исследуемой прямоугольной области.

Входные данные

В первой строке входного файла input.txt записаны два натуральных числа n и m - размеры таблицы (количество строк и столбцов соответственно). Известно, что $1 \le n, m \le 3000$. Во второй строке вначале записано одно число k – количество проникших вирусов, а далее

записаны $2k$ чисел – номера строк и столбцов первых заражённых ячеек x_i, y_i ($1 \le k \le 10$, $1 \le x_i \le n$, $1 \le y_i \le m$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – время заражения всей области.

Пример

№	input.txt	output.txt
1	4 5 2 2 1 4 5	4

Пояснения

В приведённом примере таблица имеет размер 4*5, в ней символом ‘V’ помечены проникшие вирусы. Легко посчитать, что за четыре единицы времени произойдёт заражение всей области.

V				
				V

Разбор

Для каждой ячейки перебором по всем вирусам найдем минимальное время для её заражения. Перебрав все ячейки, найдем максимум из найденных минимумов. Время заражения ячейки легко определяется так называемым манхэттенским расстоянием – суммой модулей разностей координат ячейки и вируса.

Программа на Паскале

```
var
  n, m, i, j, k, l, min, d, max : integer;
  x, y : array [1..10] of integer;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,m); read(k);
  for l:=1 to k do read(x[l],y[l]);
  max:=0;
  for i:=1 to n do for j:=1 to m do begin
    min:=n+m;
    for l:=1 to k do begin
      d:= abs(i-x[l])+abs(j-y[l]);
      if d<min then min:=d
    end;
```

```
    if min>max then max:=min
  end;
  write(max);
  close(output) end.
```

Муниципальный этап 2012-2013 уч. года, 9-11 классы

61. В автобусе

(Время - 1 сек., память - 16 Мб)

Цена проезда в автобусах нашего города — один рубль. Однако, не все так просто — каждый взрослый пассажир имеет право провезти бесплатно не более одного ребенка. Это значит, что взрослый пассажир, который провозит с собой k ($k > 0$) детей, платит всего k рублей: за один билет для себя и за (k - 1) билетов для своих детей. Также взрослый может ехать без детей, в этом случае он платит всего один рубль.

Известно, что дети не могут проезжать в автобусе без сопровождения взрослых.

Помогите посчитать минимальную и максимальную стоимость проезда в рублях, которую могли заплатить пассажиры автобуса.

Входные данные

В единственной строке входного файла input.txt записаны два целых числа n и m ($0 \le n, m \le 10^5$) — количество взрослых и количество детей в автобусе, соответственно.

Выходные данные

В единственную строку выходного файла output.txt нужно вывести:

- если в автобусе могли ехать n взрослых и m детей, то через пробел два числа — минимальную и максимальную возможную стоимость проезда этих людей, соответственно;
- в противном случае выведите «Impossible» (без кавычек).

Пример

№	input.txt	output.txt
1	1 2	2 2
2	0 5	Impossible
3	2 2	2 3

Разбор

Рассмотрим все возможные случаи. Если никто не ехал в автобусе, то необходимые величины равны нулю. Невозможен случай проезда детей без взрослых. Если не ехали дети, то каждый взрослый

платит только за себя. В остальных случаях максимальная стоимость будет тогда, когда всех детей везёт один пассажир. Минимальная стоимость определяется по разному в двух случаях. Если детей не больше взрослых, то за них не придётся платить, если каждый взрослый везёт не более одного ребёнка. В противном случае придётся платить за n взрослых и m-n детей, всего n+m-n=m рублей.

Программа на Паскале

```
var
  n, m : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(n,m);
  if (n=0) and (m=0) then write('0 0') else
    if n=0 then write('Impossible') else
      if m=0 then write(n,' ',n) else
        if m<=n then write(n,' ',n+m-1) else write(m,'
',n+m-1);
  close(output)
end.
```

62. Дроби

(Время - 1 сек., память - 16 Мб)

В то время, пока другие дети бегали по улицам или гоняли мяч, мальчик Слава сидел дома и решал сложную математическую проблему. Вкратце, проблема выглядела так.

Каждое натуральное число, начиная с трёх, можно представить в виде суммы различных натуральных чисел, например, 5=3+2=4+1. А возможно ли представить правильную дробь m/n в виде суммы различных членов гармонического ряда 1/2, 1/3, 1/4, …, то есть $m/n=1/x+1/y+1/z+\ldots$? При этом должно выполняться условие $x < y < z < \ldots$ Если существует несколько решений, то надо найти то из них, у которого значение x минимально. Если неоднозначность не снимается, то надо найти решение с минимальным y, и так далее.

Входные данные

В единственной строке входного файла input.txt записаны два натуральных числа m и n ($1 \le m < n \le 32$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести найденные числа x, y, z, ….

Пример

№	input.txt	output.txt
1	5 6	2 3

Разбор

Укажем способ вычисления x. По условию задачи из неравенства $m/n \geq 1/x$ надо взять наименьшее x. После этого имеем дробь $m/n-1/x=(m▪x-n)/(n▪x)$. Далее аналогично определяем y для найденной дроби. Вычисления прекращаем при получении дроби с числителем, равным единице. Может показаться, чтобы при этом работать с меньшими числами каждый раз рассматриваемую дробь надо сокращать, для чего числитель и знаменатель делить на их наибольший общий делитель, но численные эксперименты эту версию не подтвердили. Исследование всего диапазона исходных данных показало, что для представления результатов следует использовать целочисленный тип данных int64.

Программа на Паскале

```
var
  n, m, x : int64;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(m,n);
  while n mod m>0 do begin
    x:=(n+m-1) div m;
    write(x,' ');
    m:=m*x-n;
    n:=n*x
  end;
  write(n div m);
  close(output)
end.
```

63. Стаканы

(Время - 1 сек., память - 16 Мб)

Как известно, стакан – предмет весьма функциональный. Самый банальный способ применения – ёмкость для жидкости, самый оригинальный ещё не изобретён. А мальчик Слава строит из стаканов

башни, пользуясь удивительным свойством стаканов - ставиться друг на друга или вставляться друг в друга.

Слава строит башни из стаканов высотой 10 сантиметров, которых у него имеется бесконечное количество. Стакан можно поставить на уже имеющуюся конструкцию либо дном вниз, либо дном вверх. Если предыдущий стакан установлен аналогично новому, то конструкция вырастет на 1 сантиметр, так как стаканы надеваются друг на друга. В противном случае башня вырастет на 10 сантиметров.

Однажды Слава заметил, что ни в коем случае нельзя вставлять друг в друга более трёх стаканов, иначе один из стаканов обязательно разобьётся.

На рисунке показан пример башни высотой 32 сантиметра из 5 стаканов.

Слава умудрился построить красивую башню высотой k сантиметров. Но когда он пошёл за фотоаппаратом, чтобы запечатлеть это достижение, случайно задел конструкцию, и башня упала.

Пытаясь восстановить своё творение, Слава понял, что есть несколько способов построить башню аналогичной высоты. Помогите Славе вычислить точное количество способов.

Входные данные

В единственной строке входного файла input.txt записано натуральное число k ($1 \le k \le 100000$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – количество способов построить башню заданной высоты, взятое по модулю 1000000.

Пример

№	input.txt	output.txt
1	11	2
2	22	6
3	32	12

Разбор

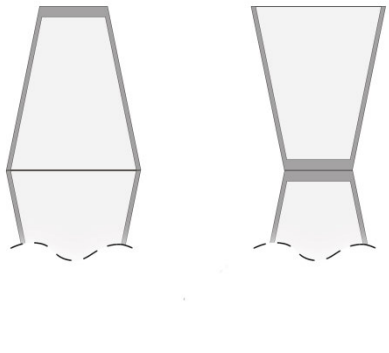
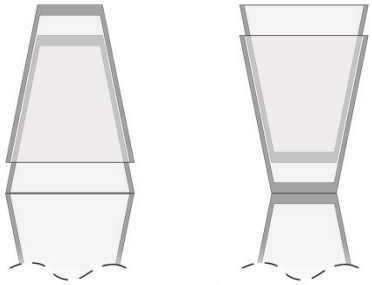
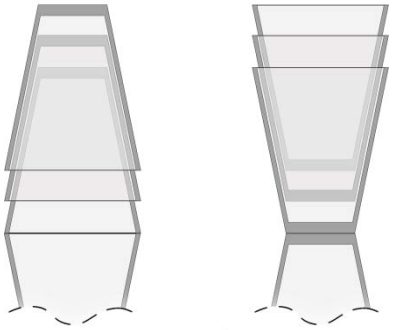
Все башни из стаканов высоты k разделим на три вида:

- к первому виду отнесём все башни, у которых самый верхний стакан ориентирован по другому, чем нижний;

- ко второму виду отнесём все башни, у которых два верхних стакана вложены друг в друга и ориентированы по другому, чем нижний;

- к третьему виду отнесём все башни, у которых три верхних стакана вложены друг в друга и ориентированы по другому, чем нижний.

Очевидно, что других башен не существует по условиям задачи. Обозначим количества башен каждого вида через a(k,1), a(k,2), a(k,3), соответственно. Надеемся, что рисунок поможет разобраться во введённых обозначениях.

		
a(k,1)	a(k,2)	a(k,3)

Если дополнить любую башню первого вида одним стаканом, вложив его в верхний или накрыв им верхний, то получим башни второго вида высоты на единицу больше. Таким образом, имеет соотношение a(k+1,2)=a(k,1). Аналогично получим, что a(k+1,3)=a(k,2). Если на любую из башен высоты k поставить стакан, сориентировав его по другому, чем верхний, то получим башню первого вида высоты k+10. Это соображение даёт соотношение a(k+10,1)=a(k,1)+ a(k,2)+a(k,3). Собрав их воедино имеем a(k,1)=a(k-10,1)+ a(k-10,2)+a(k-10,3), a(k,2)=a(k-1,1), a(k,3)=a(k-1,2). А это позволяет делать вычисления в таблице по столбцам, предварительно заполнив её первых десять столбцов нулями, кроме ячейки в первой строке и десятом столбце (в ней надо поставить двойку), так как не существует башен высоты меньше 10, и есть только две башни первого вида высоты 10.

Вид башни\k	1	2	3	4	5	6	7	8	9	10	11	12	…	k
1	0	0	0	0	0	0	0	0	0	2	0	0	…	…
2	0	0	0	0	0	0	0	0	0	0	2	0	…	…
3	0	0	0	0	0	0	0	0	0	0	0	2	…	…

Так как элементы таблицы при заданных в задаче ограничениях могут быть большими, то вычисления надо проводить в модульной арифметике.

Программа на Паскале

```
var
  k, i, j : integer;
  a : array [1..10,1..3] of longint;
  x, y, z : longint;
begin
  assign(input,'input.txt'); reset(input);
  assign(output,'output.txt'); rewrite(output);
  read(k);
  for i:=1 to 10 do for j:=1 to 3 do a[i,j]:=0;
  a[10,1]:=2;
  while k>10 do begin k:=k-1;
    x:=(a[1,1]+a[1,2]+a[1,3]) mod 1000000;
    y:=a[10,1];
    z:=a[10,2];
    for i:=1 to 9 do for j:=1 to 3 do a[i,j]:=a[i+1,j];
    a[10,1]:=x; a[10,2]:=y; a[10,3]:=z
  end;
  write((a[k,1]+a[k,2]+a[k,3]) mod 1000000);
  close(output)
end.
```

64. Вирусы - 2

(Время - 1 сек., память - 16 Мб)

Для моделирования различных объектов часто применяются так называемые клеточные поля. В простейшем случае – это прямоугольные таблицы, характеризующие некоторую область, а в каждой ячейке таблицы записывается какая-либо информация об исследуемом объекте. В биологии для моделирования распространения вирусов на плоской области в каждой ячейке помечается наличие вируса, а его распространение осуществляется в соседние ячейки по вертикали и горизонтали за одну единицу времени. Некоторые клетки обладают иммунитетом, заразить их невозможно и через них не распространяются вирусы. Напишите программу, которая найдёт минимально возможное число вирусов, с помощью которых можно заразить всю исследуемую прямоугольную область (за исключением защищённых клеток).

Входные данные

В первой строке входного файла input.txt записаны два натуральных числа n и m - размеры таблицы (количество строк и столбцов соответственно). Известно, что $1 \le n, m \le 100$. Во второй строке вначале записано одно число k – количество защищённых клеток, а далее записаны $2k$ чисел – координаты этих клеток x_i, y_i ($0 \le k \le n*m$, $1 \le x_i \le n$, $1 \le y_i \le m$).

Выходные данные

В единственную строку выходного файла output.txt нужно вывести одно число – минимально возможное число вирусов.

Пример

№	input.txt	output.txt
1	4 5 3 1 3 2 1 2 2	2

Пояснения

В приведённом примере таблица имеет размер 4*5, в ней символом ‘I’ помечены защищённые клетки. Видно, что двух вирусов достаточно для заражения всей области. Их можно поместить, например, в клетки, помеченные символом ‘V’.

V		I		
I	I			
			V	

Разбор

Заведём счётчик количества вирусов, который вначале обнулим. Просматриваем исследуемую область по слоям и при нахождении незаражённой клетки заражаем все клетки с ней связанные, а также увеличиваем счётчик на единицу. Для заражения части области вирусами используем либо рекурсивную процедуру (программа № 1), либо «волновой» алгоритм (программа № 2).

Программа № 1 на Паскале

```
var
  n, m, i, j, k, l : integer;
  a : array [0..101,0..101] of integer;
procedure Z(i,j:integer);
begin
  if a[i,j]=1 then begin
    a[i,j]:=0;
    Z(i+1,j); Z(i,j+1); Z(i-1,j); Z(i,j-1)
```

```
  end;
end;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n,m,k);
  for i:=1 to n do for j:=1 to m do a[i,j]:=1;
  for i:=1 to n do begin a[i,0]:=0; a[i,m+1]:=0 end;
  for j:=1 to m do begin a[0,j]:=0; a[n+1,j]:=0 end;
  for l:=1 to k do begin read(i,j); a[i,j]:=0 end;
  l:=0;
  for i:=1 to n do for j:=1 to m do
    if a[i,j]=1 then begin l:=l+1; Z(i,j) end;
  write(l);
  close(output)
end.
```

Программа № 2 на Паскале

```
var
  n, m, i, j, k, l, i1, j1 : integer;
  a : array [0..101,0..101] of integer;
  t : boolean;
begin
  assign(input, 'input.txt'); reset(input);
  assign(output, 'output.txt'); rewrite(output);
  read(n,m,k);
  for i:=1 to n do for j:=1 to m do a[i,j]:=0;
  for i:=1 to n do begin a[i,0]:=0; a[i,m+1]:=-1 end;
  for j:=1 to m do begin a[0,j]:=0; a[n+1,j]:=-1 end;
  for l:=1 to k do begin read(i,j); a[i,j]:=-1 end;
  l:=0;
  for i:=1 to n do for j:=1 to m do
    if a[i,j]=0 then begin
      l:=l+1; a[i,j]:=l; t:=true;
      while t do begin
        t:=false;
        for i1:=1 to n do for j1:=1 to m do begin
          if (a[i1+1,j1]=l)and(a[i1,j1]=0) then begin
t:=true; a[i1,j1]:=l end;
          if (a[i1-1,j1]=l)and(a[i1,j1]=0) then begin
t:=true; a[i1,j1]:=l end;
          if (a[i1,j1+1]=l)and(a[i1,j1]=0) then begin
t:=true; a[i1,j1]:=l end;
          if (a[i1,j1-1]=l)and(a[i1,j1]=0) then begin
t:=true; a[i1,j1]:=l end;
        end
```

```
      end;
    end;
  write(l);
  close(output)
end.
```

Анализ задач и примерная программа подготовки

В этом разделе приведен анализ рассмотренных выше задач. Для каждой из них приведена тематика и используемые алгоритмы, примерная сложность.

№	Название задачи	Тематика задачи	Используемые алгоритмы	Сложность
57	Строки в книге	Целочисленная арифметика	Действия с целыми числами	13%
30	Числа без одинаковых цифр	Целочисленная арифметика	Цифры числа	21%
13	Постоянная Капрекара	Целочисленная арифметика	Цифры числа Значение числа по цифрам	22%
44	Футбол	Целочисленная арифметика	Нахождение сумм	22%
16	Пасьянс старухи Шапокляк	Целочисленная арифметика	Действия с целыми числами	25%
19	Уравнение по основанию	Целочисленная арифметика	Значение числа по цифрам	26%
14	Короткая последовательность	Целочисленная арифметика	Построение и решение рекуррентных соотношений	28%
17	2-простое число	Целочисленная арифметика	Проверка простоты числа	28%
2	K-удивительные числа	Целочисленная арифметика	Цифры числа Значение числа по цифрам	30%
50	Степенные числа	Целочисленная арифметика	Разложение числа на простые множители	31%
8	Разные цифры	Целочисленная арифметика	Цифры числа	32%

№	Название задачи	Тематика задачи	Используемые алгоритмы	Слож-ность
48	Эволюция	Целочисленная арифметика	Действия с целыми числами	32%
5	Следующее число	Целочисленная арифметика	Цифры числа Значение числа по цифрам	36%
6	Точки отрезка	Целочисленная арифметика	Наибольший общий делитель	42%
62	Дроби	Целочисленная арифметика	Действия с целыми числами	46%
18	Арифметическая прогрессия	Математика	Вычисление арифметической прогрессии	15%
33	Количество участников олимпиады	Математика	Действия с целыми числами	17%
58	Фибоначчиева последовательность	Математика	Числа Фибоначчи	17%
42	Офис	Математика	Действия с целыми числами	18%
26	Игра со спичками	Математика	Нахождение выигрышной стратегии	18%
61	В автобусе	Математика	Действия с целыми числами	18%
54	Три грибника	Математика	Действия с целыми числами	23%
34	Садовник-художник	Математика	Правило умножения	28%
46	Турнир	Математика	Действия с целыми числами	29%
28	Сумма n-значных чисел	Математика	Сумма арифметической прогрессии	30%
43	Строительство школы	Математика	Действия с целыми числами	30%
23	Похожие массивы	Математика	Быстрая сортировка Сравнение массивов	30%

№	Название задачи	Тематика задачи	Используемые алгоритмы	Слож-ность
49	Торт	Математика	Наибольший общий делитель	32%
3	Рамка из клеток	Математика	Действия с целыми числами	33%
51	Колокол	Математика	Нахождение мини-мального элемента	33%
36	Змей Горыныч	Математика	Действия с целыми числами	34%
38	Просто простые числа	Математика	Действия с целыми числами	40%
39	Спички	Математика	Действия с целыми числами	43%
1	Сумма факториа-лов	Математика	Умножение много-значного числа на ко-роткое	45%
55	Бессмыслица	Математика	Разложения числа на слагаемые Наименьшее общее кратное	67%
41	Будильник	Математическое моделирование	Работа со временем	12%
25	Напёрстки	Математическое моделирование	Работа со строками	15%
37	Палиндромное время	Математическое моделирование	Работа со временем	24%
12	Время прибытия	Математическое моделирование	Работа со временем	26%
4	Подарки Деда Мо-роза	Математическое моделирование	Решение уравнений в целых числах	27%
22	Оптовая покупка	Математическое моделирование	Действия с целыми числами	27%
27	Шашки	Математическое моделирование	Работа со строками	27%

№	Название задачи	Тематика задачи	Используемые алгоритмы	Сложность
20	Боулинг	Математическое моделирование	Обработка массива	29%
60	Вирусы	Математическое моделирование	Нахождение минимума Нахождение максимума	30%
31	Газон	Математическое моделирование	Сокращение перебора	36%
64	Вирусы - 2	Математическое моделирование	Рекурсивный алгоритм	47%
47	Укладка плитки	Математическое моделирование	Рекурсивный алгоритм	50%
59	Сжатие бинарных последовательностей	Обработка строк	Функции работы со строками	19%
29	Детали	Обработка строк	Нахождение минимума	20%
10	Шахматный конь	Обработка строк	Кодирование символов	25%
9	Слово	Обработка строк	Числа Фибоначчи	28%
11	Химическая формула	Обработка строк	Обработка строк	36%
53	Баланс скобок	Обработка строк	Реализация стека	42%
35	Абракадабра	Динамическое программирование	Построение и решение рекуррентных соотношений	31%
40	Числа	Динамическое программирование	Построение и решение рекуррентных соотношений	35%
32	Выбор приборов	Динамическое программирование	Построение и решение рекуррентных соотношений	37%
7	Коридор	Динамическое программирование	Построение и решение рекуррентных соотношений	38%

№	Название задачи	Тематика задачи	Используемые алгоритмы	Сложность
63	Стаканы	Динамическое программирование	Построение и решение рекуррентных соотношений	40%
21	Земельный комитет	Динамическое программирование	Построение и решение рекуррентных соотношений	41%
24	Роман в томах	Динамическое программирование	Построение и решение рекуррентных соотношений	41%
45	Две цифры	Динамическое программирование	Построение и решение рекуррентных соотношений	42%
15	Последовательности из 0 и 1	Динамическое программирование	Построение и решение рекуррентных соотношений Длинная арифметика	47%
52	Ленточка	Динамическое программирование	Построение и решение рекуррентных соотношений	60%
56	Ленточка - 2	Динамическое программирование	Построение и решение рекуррентных соотношений	65%

В итоге получается пятнадцать задач на целочисленную арифметику, двадцать задач связаны с базовыми математическими понятиями и двенадцать с математическим моделированием. В шести задачах надо было использовать работу со строками, а ещё одиннадцать задач используют при решении подход, называемый динамическим программированием.

Проведенный анализ задач позволяет предложить следующую **примерную программу подготовки учащихся** к школьному и муниципальному этапам олимпиады школьников по информатике.

1. ***Целочисленная арифметика***
   - Действия с целыми числами: сложение, вычитание, умножение, остаток и целая часть от деления
   - Нахождение цифр числа

- Нахождение значения числа по цифрам (схема Горнера)
- Нахождение степени числа

2. *Базовые математические понятия и алгоритмы*

- Нахождение сумм и произведений
- Нахождение минимума и максимума
- Простое число, разложение числа на простые множители
- Наибольший общий делитель, алгоритм Евклида, наименьшее общее кратное
- Арифметическая и геометрическая прогрессии
- Числа Фибоначчи
- Рекуррентные соотношения
- Системы счисления
- Правила сложения и умножения в комбинаторике
- Перестановки, размещения, сочетания. Подмножества
- Понятие рекурсии
- Длина отрезка, площадь треугольника
- Простейшие стратегии игры

3. *Математическое моделирование*

- Работа с временем
- Работа с датами
- Составление линейных уравнений и их решение в целых числах
- Обработка одномерных и двумерных массивов
- Сортировка массивов
- Длинная арифметика
- Понятие стека, очереди

4. *Обработка строк*

- Кодирование символов
- Подпрограммы работы с символами и строками

5. *Динамическое программирование*

- Построение одномерных рекуррентных соотношений
- Нахождение решений одномерных рекуррентных соотношений
- Построение двумерных рекуррентных соотношений

- Нахождение решений двумерных рекуррентных соотношений

Организация и проведение тренировок

В этом разделе приведено описание интерфейсов участника и преподавателя для организации и проведения на сайте «Интернет-олимпиады для школьников Ханты-Мансийского автономного округа – Югры» тренировок школьников на рассмотренных задачах.

Интерфейс участника

В левом верхнем углу первой страницы сайта находится панель «Вход», состоящая из полей ввода логина (имени пользователя на сайте) и пароля, а также ссылок на страницы регистрации и восстановления пароля. Данная панель отображается только в том случае, если пользователь не прошел процедуру аутентификации (т.е. не ввел свои логин и пароль).

Вход (аутентификация)

Для входа на сайт необходимо ввести имеющиеся логин и пароль в соответствующие поля ввода в панели «Вход» и нажать кнопку «Войти». Если их нет, то надо пройти регистрацию.

Регистрация

Для регистрации необходимо в нижней части панели «Вход» нажать на ссылку «Регистрация». На появившейся странице необходимо заполнить все поля желаемое имя пользователя (логин), пароль, адрес электронной почты и т.д. После заполнения полей и нажатия кнопки «Отправить» происходит мгновенная регистрация, после чего можно процедуру аутентификации. В случае наличия каких-либо ошибок при заполнении формы регистрации система сообщит вам об этом.

Восстановление пароля

В том случае, если вы уже были зарегистрированы на сайте, но забыли пароль, то есть возможность восстановления пароля. Для этого нужно нажать на ссылку «Забыли пароль?». На появившейся странице необходимо ввести логин, для которого нужно восстановить пароль. При нажатии на кнопку «Отправить» происходит отправка письма на адрес электронной почты, указанный при регистрации. В письме будет находиться ссылка на страницу сброса пароля.

Далее все процессы работы с сайтом описываются при наличии авторизации.

Участие

В левой части страницы отображается список соревнований. Для выбора интересующего вас соревнования необходимо нажать на ссылку с его названием. На появившейся странице отобразится информация о соревновании — статус, планируемые дата и время начала, продолжительность и т. д. В случае, если вам разрешено участие в данном соревновании, то будет отображаться ссылка «Подать заявку», при нажатии на которую вы подтвердите свое желание участвовать в соревновании. Если вместо ссылки «Подать заявку» отображается надпись «Регистрация закрыта», то это значит, что либо соревнование не предназначено для свободного участия, либо истек разрешенный срок регистрации.

Если на данной странице отображается кнопка «Участвовать», то вы можете принимать участие в решении задач соревнования.

При нажатии на кнопку «Участвовать» открывается страница участия в соревновании.

В левой части данной страницы «Соревнование» отображается список задач соревнования. Если соревнование еще не началось, то данный список скрыт. При нажатии на ссылку с названием задачи открывается страница с условием задачи.

В нижней части страницы с описанием задачи находится панель отправки решения (ответа на задачу) на проверку. В зависимости от типа задачи данная панель будет выглядеть по-разному.

Если задача имеет тип «Выбор одного варианта из нескольких», то отображается список вариантов ответов, из которых вам нужно выбрать правильный, после чего нажать на кнопку «Отправить».

Если задача имеет тип «Короткий ответ», то отображается небольшое текстовое поле, в которое вам нужно вписать краткий ответ на вопрос, после чего нажать на кнопку «Отправить».

Если задача имеет тип стандартной задачи на программирование, то отображается поля выбора языка программирования и загрузки файла на сервер. Для отправки решения на проверку необходимо выбрать файл с исходными текстами проверяемой программы и указать язык программирования, на котором данная программа написана.

После проверки задачи в нижней части страницы отображается таблица с результатами проверки.

Для отображения страницы с таблицей результатов соревнования нужно нажать на ссылку «Рейтинг» в верхней части страницы.

Общий список задач соревнования отображается при нажатии на ссылку «Задачи».

Интерфейс преподавателя

Для того, чтобы преподавателю можно было организовать и провести своё соревнование надо иметь логин и пароль для входа на сайт. Как их получить описано выше в разделе «Интерфейс участника». После этого на электронный адрес авторов пособия (aav@uriit.ru) надо направить электронное письмо с просьбой о предоставлении доступа к проведению соревнований.

После получения доступа в левой части страниц появится панель «Меню преподавателя», находящаяся ниже панели «Вход»/«Кабинет». При нажатии на ссылку «Мои соревнования» в панели «Меню преподавателя» откроется страница с перечислением соревнований, к которым у вас есть доступ для настройки соревнований (т. е. к которым вам его дала администрация сайта). Справа от названия каждого соревнования на данной странице находится ссылка [P], при нажатии на которую открывается страница редактирования базовых настроек соревнования.

На странице редактирования настроек соревнования для изменения доступны следующие параметры:

- название соревнования;
- система правил (acm – студенческие, olympiad – школьные, kirov - смешанные);
- дата и время начала соревнования;
- длительность (в минутах).

Также есть возможность настройки видимости соревнования в списке («Показать в разделе»).

Для настройки списка задач соревнования используется раздел «Задачи» в нижней части страницы. В данный момент реализовано добавление задач из сборников, в том числе «Задачи муниципального этапа». Для каждой задачи в списке задач соревнования есть возможность указания баллов за полное решение и штрафных баллов за отправку неправильного решения.

Литературные и интернет-источники

Далее приведены литературные и интернет источники, которые авторы рекомендуют использовать в подготовке к олимпиадам. Из большого объёма имеющейся литературы приведены источники, в которых изложение материала достаточно элементарно, чтобы было понятно начинающим, но в них содержатся и сведения для успешной подготовки к олимпиадам регионального и заключительного этапов. Приведённые в списке интернет-ресурсы рекомендуется использовать для аудиторной и/или самостоятельной подготовки. Кроме того, советуем участвовать в проводимых на них тренировках и соревнованиях.

1. *Алексеев А.В., Беляев С.Н.* Подготовка школьников к олимпиадам по информатике с использованием веб-сайта: Учебно-методическое пособие для учащихся 7-11 классов. – Ханты-Мансийск: РИО ИРО, 2008. – 284 с.

2. *Андреева Е.В. и др.* Математические основы информатики. Элективный курс [Текст] /Е.В. Андреева и др.:. – М.: Бином. Лаборатория знаний, 2005. – 328 с.

3. Дистанционная подготовка по информатике [Электронный ресурс]. http://informatics.mccme.ru

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

5. Интернет-портал организационно-методического обеспечения дистанционных олимпиад по программированию для одаренной молодежи учебных заведений Украины [Электронный ресурс]. http://www.e-olimp.com/

6. *Кирюхин В.М.* Методика проведения и подготовки к участию в олимпиадах по информатике [Текст]: всероссийская олимпиада школьников /В.М. Кирюхин. –2-е изд. – М.: БИНОМ. Лаборатория знаний, 2012. – 271 с. [4] л. ил. : ил. – ISBN 978-5-9963-0731-9.

7. *Окулов С.М.* Дискретная математика. Теория и практика решения задач по информатике [Текст]: : учебное пособие / С.М. Окулов. – М.: БИНОМ. Лаборатория знаний, 2011, 288 – 422 с.

8. *Окулов С.М.* Основы программирования. – М.: БИНОМ. Лаборатория знаний, 2012. – 336 с.

9. *Окулов С.М.* Программирование в алгоритмах [Текст]: – М.: БИНОМ. Лаборатория знаний, 2002. – 341 с.

10. Олимпиады по информатике, ХМАО-Югра [Электронный ресурс]. http://www.acmu.ru/

11. *Фаронов В.В.* Турбо Паскаль 7.0. Практика программирования [Текст] / В.В. Фаронов – М.: Нолидж, 1997. – 616 с.

12. *Волчёнков С.Г. и др.* Ярославские олимпиады по информатике. [Текст] Сборник задач с решениями / С.Г. Волчёнков и др. – М.: БИНОМ. Лаборатория знаний, 2010. – 405 с.

Содержание

Алексеев Александр Владимирович, главный научный трудник автономного учреждения Ханты-Мансийского автономного округа – Югры «Югорский научно-исследовательский институт информационных технологий», доцент кафедры компьютерного моделирования и информационных технологий Института систем управления и информационных технологий федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Югорский государственный университет», кандидат педагогических наук, доцент.

В 1976 году закончил механико-математический факультет Новосибирского государственного университета. До 1988 года работал в Вычислительном центре Сибирского отделения АН СССР в г. Красноярске. С 1986 года связан со школьной информатикой, с 1988 по 2001 год работал в системе школьного образования Красноярского края. С этого же времени занимается олимпиадной информатикой, организатор и составитель заданий для первых районных, краевых олимпиад школьников. С 1989 года член жюри Всероссийских олимпиад школьников по информатике. В 1998 году защитил кандидатскую диссертацию по внеклассной работе со школьниками по информатике. С 2002 года работает в образовательных учреждениях Ханты-Мансийского автономного округа. С 2006 по 2008 годы, находясь в должности директора Центра информатизации образования бюджетного образовательного учреждения дополнительного профессионального образования Ханты-Мансийского автономного округа – Югры «Институт развития образования», Александр Владимирович стал инициатором проведения муниципального и регионального этапов всероссийской олимпиады школьников по информатике в дистанционной форме, в 2010 – инициатором проведения заключительного этапа олимпиады по информатике в г. Ханты-Мансийске. С 2006 года он является председателем жюри регионального этапа, председателем региональной предметно-методической комиссии по разработке олимпиадных заданий регионального, а с 2009 года – муниципального этапов. На заключительном этапе всероссийской олимпиады школьников по информатике 2012-2013 уч. года в г. Казани А.В. Алексеев награждён дипломом и памятным знаком «25 лет Всероссийской олимпиады школьников по информатике».

Карелин Виталий Александрович, аспирант федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Югорский государственный университет».

В 2005 закончил Югорскую физико-математическую школу, в 2010 – Югорский государственный университет, заканчивает аспирантуру. В олимпиадной информатике с 2003 года как участник олимпиад. Дипломная работа и готовящая кандидатская диссертация посвящены автоматизации проведения соревнований школьников. Участник полуфинальных соревнований студенческого чемпионата мира по программированию. С 2008 года входит в состав жюри регионального этапа, осуществляет функционирование автоматизированной проверяющей системы проведения соревнований школьников. Награждён грамотами Департамента образования и молодёжной политики Ханты-Мансийского автономного округа – Югры, Департамента образования, культуры и молодёжной политики Чукотского АО.

Короткова Екатерина Михайловна, ведущий программист автономного учреждения Ханты-Мансийского автономного округа – Югры «Югорский научно-исследовательский институт информационных технологий», В 2006 закончила Югорскую физико-математическую школу, в 2011 – механико-математический факультет Новосибирского государственного университета. Кандидат в мастера спорта по шахматам. Занималась олимпиадной подготовкой школьников Летней профильной школы.

Алексеев Александр Владимирович
Карелин Виталий Александрович
Короткова Екатерина Михайловна

Олимпиады по информатике
в Ханты-Мансийском автономном округе – Югре (2008 – 2013 гг.):
сборник олимпиадных заданий муниципального этапа всероссийской олимпиады школьников для учащихся 7-11 классов

Оригинал-макет изготовлен РИО ИРО

Технический редактор:
Семёнова В.В.
Дизайн обложки:
Белов М.В.

Подписано в печать 07.06.2013.
Формат 60*84/16. Гарнитура Times New Roman.
Усл.п.л. 6,7. Заказ № 276. Тираж 200 экз.

Институт развития образования

Ханты-Мансийский автономный округ – Югра
628012, г. Ханты-Мансийск, ул. Чехова, 12