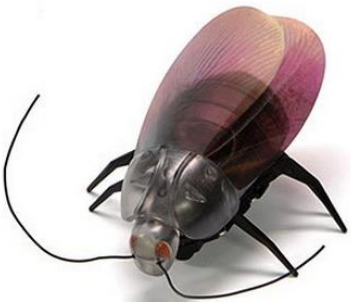
Оглавление

7-8 класс

Задача 1. Робот Gokiji

Имя входного файла:	Gokiji.in
Имя выходного файла:	Gokiji.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб



В настоящее время наблюдается бум в производстве интерактивных игрушек-роботов. Сразу вспоминаются роботы-собаки, широко распространенные в Японии, но это не всё. Существуют роботы-кошки, лошади, пингвины, кенгуру и даже тюлени.

И, конечно, не остались без внимания насекомые, особенно тараканы. Японская компания предлагает домашних питомцев нового поколения – тараканов-роботов Gokiji. У них есть усы, крылья, лапки и лампочка, чтобы светить в темноте. Gokiji слушаются пульта управления, бегают, шевелят усами, крутятся на месте, копошатся в кучке бумаг.

Мальчик Вова получил от родителей такую игрушку. Ему понравилось запускать таракана по комнате. Вова заметил, что у Gokiji есть интересный режим движения: он идет со скоростью v см/с и каждые L см останавливается, чтобы в течение K секунд охладить свое тело взмахами крыльев. Управлять тараканом можно с помощью пульта. Также можно получать информацию о параметрах движения робота в специальном приложении, установленном на смартфон Вовы.

Через какое-то время Вова понял, что если знать все вышеперечисленные параметры движения Gokiji и запускать его по диагонали, то можно определить площадь любой квадратной комнаты.

От Вас **требуется** написать программу расчета площади квадратного помещения по указанным параметрам движения таракана Gokiji по диагонали этого помещения.

Формат входного файла

В первой строке входного файла записаны 5 вещественных чисел:

число L см – расстояние, через которое Gokiji останавливается ($0<L\leq 35$);

число v см/с – скорость Gokiji ($5 \leq v < 105$);

число K с – время на обмахивание крыльями ($5 \leq K < 105$);

число S с – время, через которое Gokiji добегает до противоположного угла ($5 \leq S < 10000$).

Формат выходного файла

В первой строке входного файла записано одно вещественное число с точностью до двух знаков после запятой – площадь участка в метрах.

Примеры

Gokiji.in	Gokiji.out
10.5 3 5 224.29	4.00

Gokiji.in	Gokiji.out
21.5 9 9 218.13	9.00

Решение.

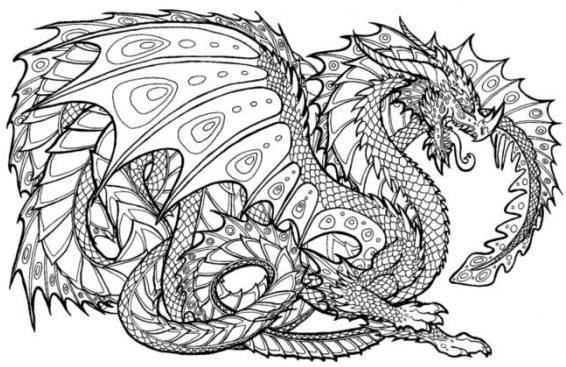
Решение задачи сводится к нахождению квадрата длины стороны квадратного помещения. Находим время на 1 период, состоящий из пробега и остановки (P). Затем определяем количество полных периодов (kp) и расстояние, пройденное за полные периоды (d). Затем находим остаток времени, который не составляет полный период (T). Умножаем его на скорость и получаем остаток диагонали (d2). Находим длину диагонали (d3), возводим ее в квадрат. Это квадрат равен сумме квадратов катетов, т.е. сумме квадратов сторон квадратного помещения. Делим эту величину пополам и получаем сторону помещения в квадрате, т.е. площадь помещения (d3). Поскольку весь расчет шел в сантиметрах, то при выводе ответа переводим его в квадратные метры.

Таким образом, решение может принять следующую форму.

```
var L,v,K,S,P,kp,d,d2,d3,T : real;
begin
  assign(input,'Gokiji.in');
  reset(input);
  readln(L,v,K,S);//L-расстояние, через которое остановка
                  //v-скорость,K-время обмахивания крыльями
                  //S - полное время пробега диагонали
  close(input);
  P:=(L/v+k);     //длина периода в сек
  kp:=trunc(S/P); //кол-во полных периодов
  d:= kp*L;       //расстояние на все полные периоды
  T:=S-kp*P;      //остаток времени, который без остановки
  d2:=v*T;        //остаток пути T на скорость
  d3:=d+d2;       //длина диагонали
  d3:=(d3*d3)/2;  //в квадрате диагонали - сумма
                  // квадратов двух равных катетов
  assign(output,'Gokiji.out');
  rewrite(output);
  writeln(d3/10000:0:2);
  close(output);
end.
```

Задача 2. Змей Горыныч

Имя входного файла:	apples.in
Имя выходного файла:	apples.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб



Как известно всем жителям нашей страны с детства, трехголовый Змей Горыныч немало бед доставил Князю Киевскому. Вот и опять идет он войной на Киев-град. Послал князь за сильномогучими богатырями-защитниками, но оказалось, что их нет в Киеве, собирают они дань с хана Бекета. Шлет князь гонца к богатырям, но нужно время, чтобы прибыли они на бой со Змеем Горынычем. Надо задержать Змея Горыныча. К счастью, есть у князя секретный прием на этот случай. Секретный прием основан на том, что змей Горыныч очень любит яблоки. Как только он их видит, забывает про все на свете и пока всё не съест, даже на войну не отвлекается.

Поэтому есть у князя в запасе три огромные чаши каменные. Выставляют их киевляне перед главными воротами города и всем миром насыпают в них яблоки румяные, пахучие и сладкие. В левой чаше оказывается A яблок, в средней вазе – B яблок, в правой – C яблок.

И вот как только подлетает Горыныч к Киеву, чует он яблоки замечательные и сразу к ним сворачивает. Приземляется он на поляну перед чашами и давай яблоки есть: одно берет их из левой чаши, другое – из средней чаши, третье – из правой, потом опять из средней, потом опять из левой, средней, правой, средней и т. д. (слева направо, затем налево, опять направо и т.д.). Съедает он любое яблоко на одинаковое время.

И все бы хорошо, но если Змей Горыныч хочет взять яблоко из какой-то вазы, а яблоки там закончились, то он сильно огорчается, в другие вазы уже не заглядывает (очень капризная и обидчивая рептилия!) и сразу нападает на Киев-град.

От Вас **требуется** определить, сколько времени есть в запасе у князя Киевского и киевлян до того момента, как нападет Змей на город.

Формат входного файла

В первой строке записаны три натуральных числа A, B, C – количество яблок в левой, средней, правой чаше. Сумма трёх данных чисел не превосходит 2×10^9.

В следующей строке находится натуральное число $3 \le t \le 60$ – время в минутах, которое тратит Змей на то, чтобы съесть одно яблоко.

Формат выходного файла

В первой строке входного файла записано время, через которое змей нападет на Киев-град. Формат вывода времени – t:mm, где t – количество часов, mm – количество минут, дополненное слева нулем до двух разрядов (при необходимости).

Примеры

apples.in	apples.out
3 3 3	0:07

1	

Решение.

Решение, полностью моделирующее процесс поедания яблок, то есть уменьшающее их число на 1 в каждой вазе в соответствии с описанным в условии алгоритмом, не будет укладываться в ограничение по времени работы программы.

Для получения полного решения заметим, что процесс взятия яблок содержит цикл «левая ваза, средняя ваза, правая ваза, средняя ваза», который затем повторяется. За один проход такого цикла число A уменьшается на 1, число B уменьшается на 2, число C уменьшается на 1. Посчитаем, сколько раз будет выполнен цикл — это минимум из чисел A, [B / 2] и C (под записью [B / 2] подразумевается целая часть от деления B на 2, то есть операция целочисленного деления). Запишем количество проходов цикла в переменную k и уменьшим значение переменных A и C на k, а значение переменной B на 2*k. За k исполнений цикла суммарно будет съедено 4*k яблок.

Следующий проход цикла не будет выполнен полностью. Посмотрим на значение переменных A, B, C в том порядке, в котором берутся яблоки из соответствующих ваз. Если A = 0, то нельзя на следующем шаге взять яблоко из первой вазы, и ответом будет 4*k. Если B = 0, то будет взято яблоко из первой вазы, но во второй вазе яблоки закончились, поэтому ответ будет 4*k + 1. Если же C = 0, то, аналогично, можно взять еще два яблока из левой и средней вазы, и ответ будет 4*k + 2. Наконец, если все эти условия не выполнены, то ответ будет 4*k + 3.

Пример решения задачи:

```
var a,b,c,t,k,tt: integer;
begin
  assign(input,'apples.in');
  reset(input);
  readln(a,b,c);
  readln(t);
  close(input);
  k := min (a,b div 2); //min(a, b div 2, c);
  k:= min (k,c);
  a := a - k;
  b := b - 2 * k;
  c := c - k;
  assign(output, 'apples.out');
  rewrite(output);
  if a = 0
  then  tt := 4 * k * t //в минутах
  else if b = 0
     then tt := (4 * k + 1) * t
     else if c = 0
          then tt := (4 * k + 2) * t
          else tt := (4 * k + 3) * t;
```

```
  c := tt mod 60;
  tt := tt div 60;
  if c<10
  then write(tt,':0',c)
  else write(tt,':',c);
  close(output);
  end;
end.
```

Задача 3. Камни Данилы-мастера

Имя входного файла:	malachite.in
Имя выходного файла:	malachite.out
Максимальное время работы на одном тесте:	1 секунда
Максимальный объем используемой памяти:	64 Кб

С детских лет всем жителям нашей страны известны сказы Павла Петровича Бажова. Герой сказа «Каменный цветок», Данила-мастер, живет в очень тяжёлых условиях. Но он очень трудолюбив, любит своё ремесло и стремиться к тому, «чтобы полную силу камня самому поглядеть и людям показать».

Не раз он в лес и на заброшенный рудник ходил, искал камни малахитовые для своих поделок. Присмотрит камень подходящий, положит в карман, присмотрит еще – положит в другой. А карманов у него всего два. И когда они заполняются, приходится для нового, более подходящего камня, освобождать один из карманов, выбрасывая из ранее выбранных камней наименее подходящий. Оценивает камни Данила-мастер коэффициентом от 1 до 1 000 000, причем, чем больше коэффициент, тем ценнее камень.

Как Вы помните, Медной горы хозяйка помогает Данила-мастеру. Она подбрасывает ему много разных камней, чтобы он мог как можно лучше реализовать свой замысел. Но в этом и проблема – камней слишком много.

От Вас **требуется** написать программу, которая определяет самые лучшие 2 камней, которые Данила-мастер выберет и принесет домой.

Формат входного файла

В первой строке записано целое число $4<N<1\,000\,000$ – количество камней, которые Медной горы хозяйка подбросила Данила-мастеру.

В следующих N строках находятся N целых чисел $0 \le x_i < 1000000$, характеризующих степень соответствия i-го камня замыслу Данилы-мастера.

Формат выходного файла

В первой строке входного файла записаны через пробел два целых числа – характеристики двух самых лучших, с точки зрения Данилы-мастера, камней.

Примеры

malachite.in	malachite.out
8 10 44 250 103 505 520 60 17	520 505

Решение.

Решение задачи сводится к нахождению двойного максимума заданной последовательности целых чисел. Для хранения этих максимумов удобно использовать 2 переменные, значения которых упорядочены по убыванию, т.е. в первом самое большое число, а во втором – число поменьше.

Очередное входное число сравниваем с первым максимумом. Если оно больше, то первый максимум перекладываем во второй, а новое число кладем в первый максимум. Если новое число не больше первого максимума, но больше второго, то помещаем его во второй максимум. Так мы не нарушаем упорядоченности максимумов. Если число меньше второго максимума, то не делаем ничего.

Таким образом, решение может принять следующую форму.

```
var
   N,i,j,x,t,h :integer;
   A1, a2: integer; // максимумы
begin
  assign(input,'malachite.in');
  reset(input);
  readln(N);
  readln(a1);
  readln(a2);
  if a2>a1 then swap(a1,a2);
  for i:=3 to N do
  begin
    readln(x); //очередное число
    if a1<x then begin a2:=a1; a1:=x end
    else if a2<x then a2:=x;
  end;
  close (input);
  assign (output,'malachite.out');
  rewrite(output);
  write(a1,' ',a2);
end.
```

Задача 2. Домино Колывана

Имя входного файла: Domino.in
Имя выходного файла: Domino.out

Максимальное время работы на одном тесте: 1 секунда
Максимальный объем используемой памяти: 64 Кб

Как известно всем жителям нашей страны, во времена Ильи Муромца, Добрыни Никитича и Алеши Поповича жил в стольном граде Киеве хитрющий купец Колыван. Немало бед принес он народу киевскому, немало проблем доставил князю. Но, как помните, был у него талант: в какую игру не сядет играть, всегда выигрывает. И никто ему противиться не мог, знают что проиграют, а все равно соглашаются играть с Колываном.

И вот в очередной раз сел князь Киевский и хитрец Колыван в домино играть. Знает князь, что нельзя играть, а все равно согласился и даже половину казны своей на кон поставил.

Играет Колыван, жульничает, неправильно кости домино на стол выкладывает, а князь-то так ничего и не замечает! И вот игра закончена, Колыван уже руки свои загребущие к казне потянул, а сам ухмыляется! И тут выбегает конь говорящий Юлий и кричит:

- Люди добрые, да что же это делается! Мне, главному библиотекарю князя Киевского, жалование еще не заплачено, а этот упырь уже руки свои поганые к казне государственной тянет! И князя запутал-заморочил, и всему окружению государеву пыль в глаза пустил, не видим мы, где Колыван нажульничал! Кто поможет нам казну спасти, Колывана на чистую воду вывести? Кто сможет проверить, что последовательность костяшек домино на столе выложена неправильно?»

Итак, **требуется** написать программу, которая поможет выяснить, является ли последовательность из N костяшек домино ($2 \le N \le 22$) выложенной правильно.

Формат входного файла

В первой строке записано одно натуральное число $2 \le N \le 22$ – количество костяшек домино на столе;

В последующих N строках записано по одному неотрицательному целому числу $0 \le x_i \le 66$, кодирующему кость домино следующим образом: число от 0 до 66, например, 24 это кость, на которой 2 и 4, 33 – это 3 и 3, 5 – это 0 и 5 и т.д.

Формат выходного файла

В единственной строке выходного файла содержится единственная строка, в которой записано слово YES, если условие выполнено или слово NO, иначе.

Примеры

Domino.in	Domino.out
7 32 22 25 53 33	YES

36 63	

Domino.in	Domino.out
6 15 43 34 40 6 66	NO

Решение.

Решение основано на разборе чисел, кодирующих костяшки домино, на цифры. Пример реализации:

```
var
     i,x,x1,ml1,st,ml,n:integer;
begin
     assign(input,'domino.in');
     reset(input);
     assign(output,'domino.out');
     rewrite(output);
     readln(n);
     readln(x1);
     ml1:=x1 mod 10;
     for i:=2 to n do
     begin
          readln(x);
          st:=x div 10;
          ml:=x mod 10;
          if ml1<>st then
          begin
               write('NO');
               close(output);
               halt;
          end
          else ml1:=ml;
     end;
     write('YES');
     close(output);
end.
```