Оглавление

9-11 класс

Задача 1. Робот Венера

Имя входного файла:	Robocat.in
Имя выходного файла:	Robocat.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-такараны, стрекозы, лошади, пингвины, кенгуру и даже тюлени.

Конечно, не остались без внимания и кошки. Робот-кошка по имени Венера выглядит, как настоящая живая кошка и ведет себя точно также. Она грациозно и уверенно движется, умеет мяукать, мурчать, реагирует на слова и поглаживания, а также валяется, выгнувшись на спине, демонстрируя свое хорошее расположение духа. Кроме того, кошка обладает способностью к обучению и может совершенствовать свои кошачьи навыки.

Девочке Кате, которая живет в загородном доме, как раз подарили на день рождения такую кошку Венеру. Игрушка, конечно, очень понравилась. Катя даже стала выводить кошку на прогулку. Через какое-то время Катя заметила, что Венера, выходя на прогулку, стала всегда в начале прогулки делать одно и то же – обходить по периметру весь квадратный земельный участок, на котором стоит дом Кати.

При этом ведет себя Венера достаточно интересно: она все время держится на расстоянии N м от забора, идет медленно, со скоростью v см/с, и каждые L м останавливается, чтобы в течение K секунд чистить лапы. К месту начала обхода она возвращается через S секунд.

Подумав, Каты поняла, что можно определить площадь любого квадратного участка, если знать все вышеперечисленные параметры движения робота-кошки.

От Вас **требуется** помочь Кате, написав программу расчета площади квадратного участка по указанным параметрам прогулки робота Венеры.

Формат входного файла

В первой строке входного файла записаны 5 вещественных чисел:

число N м – расстояние, на котором держится от забора Венера ($0.5 \le N \le 30$);

число L м – расстояние, через которое Венера останавливается ($0<L\le 35$);

число v см/сек – скорость Венеры ($5 \le v <105$);

число K сек – время на очистку лапок ($5 \le K < 105$);

число S сек – время, через которое Венера возвращается к месту начала обхода ($5 \le S < 10000$).

Формат выходного файла

В первой строке входного файла записано одно вещественное число с точностью до двух знаков после запятой – площадь участка м^2.

Примеры

Robocat.in	Robocat.out
1.5 3 85 6 212	400.00

Robocat.in	Robocat.out
2.0 1.75 25 60 1000	196.00

Решение.

Сначала найдем путь, который проходит кошка за время обхода.

Кошка тратит t_1 на то, чтобы пройти L метров. Она чистит лапы в течение K секунд. Значит, она совершает один L-метровый переход каждые $t_1 = L/v + K$ секунд.

За время обхода кошка совершила несколько полных L-метровых переходов; кроме того, она прошла еще последний отрезок пути, который мог оказаться и короче, чем L метров.

Найдем, сколько полных L-метровых переходов совершила кошка. Разделим время обхода на время полного L-метрового перехода вместе с чисткой лапок и возьмем целую часть $t_2 = [\ s/t_1\ ]$. Это значит, что кошка совершила t_2 полных L-метровых переходов и еще могла пройти некоторое расстояние.

На чистку лап кошка потратила $t_3 = t_2 \cdot k$

Оставшееся чистое время движения равно $(S - t_3)$, а его длина $(S - t_3) \cdot v$.

По условию кошка ходит внутри квадрата на расстоянии N м от его сторон. Это означает, что она ходит по квадрату со стороной на $2N$ м короче, чем сторона садового участка. Длина стороны квадрата по которому ходит кошка составляет (s-t3)*v/4 м, добавив к этой величине 2*n и переведя все величины в метры получим сторону участка d2. Площадь квадратного участка, по которому ходит кошка, равна d2*d2.

Таким образом, решение может принять следующую форму:

```
var n,l,v,k,s,t1,t2,t3,d1,d2,d3:real;
begin
  assign(output,'robocat.in');
  reset(input);
  readln(n,l,v,k,s);
  t1:=((l*100)/v+k);
  t2:=trunc(s/t1);
```

```
  t3:= t2*k;
  d2:=((s-t3)*v/4+2*n*100)/100;
  d3:=d2*d2;
  close(input);
  assign(output, 'robocat.out');
  rewrite(output);}
  writeln(d3:2:2);
  close(output);
end.
```

Задача 2. Данила-мастер

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 пять. И когда они заполняются, приходится для нового, более подходящего камня, освобождать один из карманов, выбрасывая из ранее выбранных пяти 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 5 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 пять целых чисел – характеристики пяти самых лучших, с точки зрения Данилы-мастера, камней.

Примеры

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

Решение.

Решение задачи сводится к нахождению пяти последовательных максимумов во входном потоке целых чисел. Для хранения этих максимумов удобно использовать массив из 5 элементов, которые упорядочены по убыванию, т.е. на первом месте в массиве лежит самое большое число, потом поменьше и так далее.

Для очередного входного числа ищем место в массиве. Если это место не равно 6, то сдвигаем хвост массива вправо на одну позицию и помещаем элемент в массив. Так мы не нарушаем его упорядоченности.

Таким образом, решение может принять следующую форму.

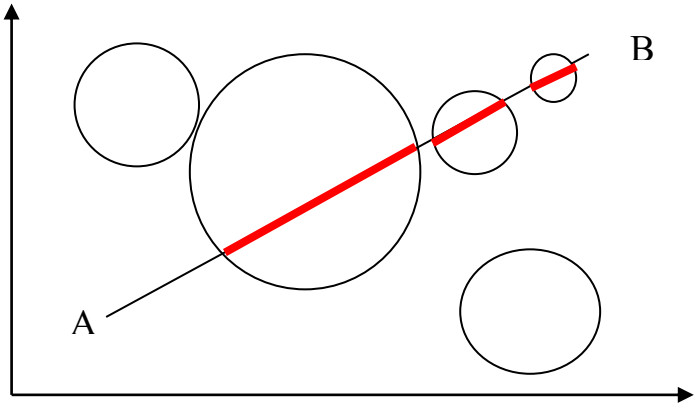
```
var
   N,i,j,x,t,h :integer;
   a:array [1..5] of integer; //массив для максимумов
begin
  for i:=1 to 5 do a[i]:=0;
  assign(input,'malachite.in');
  reset(input);
  readln(N);
  readln(a[1]);
  t:=1; //заполненное количество максимумов
  for i:=2 to N do
  begin
    readln(x); //очередное число
    h:=6;    //поиск позиции очередного числа в массиве a
    for j:=1 to h-1 do
    if a[j]<=x then begin h:=j; break end;
    if h<>6 then
    begin  //сдвиг массива a
      for j:=4 downto h do a[j+1]:=a[j];
      a[h]:=x
    end;
  end;
  close (input);
  assign (output,'malachite.out');
  rewrite(output);
  write(a[1],' ',a[2],' ',a[3],' ',a[4],' ',a[5]);
end.
```

Задача 3. Безопасность полета

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

Для обеспечения безопасности полетов над территорией некоторой страны были установлены N радаров, наблюдающих воздушную обстановку. Области действия радаров круговые и не пересекаются. Самолет пролетает над территорией этой страны по прямолинейной траектории, из точки A в точку B.

Требуется написать программу, определяющую, сколько километров траектории полета (с точностью до второго десятичного знака) самолет находился в областях действия радаров (на рисунке эти участки траектории выделены жирными отрезками).



Формат входного файла

В первой строке вещественные числа x_A, y_A, x_B, y_B ($-1000 \le x_A, y_A, x_B, y_B \le 1000$) – координаты точек A и B. Во второй строке число N ($1 \le N \le 1000$) количество радаров.

Далее N строк по 3 вещественных числа – x_k, y_k – координаты расположения очередного радара, R_k – радиус его действия ($-1000 \le x_k, y_k, R_k \le 1000$, $1 \le k \le N$).

Формат выходного файла

Одно вещественное число, с округлением до двух десятичных знаков, – расстояние, которое пролетит самолет в областях действия радаров.

Примеры

distance.in	distance.out
1 7.5 9 3.5 3 4 2 1 5 5.5 2 7.5 3.5 1	5.48
distance.in	distance.out
1 5 2 5	0.00

1 4 4 1	

Решение.

Решение задачи сводится к решению следующей подзадачи. Дана окружность, заданная координатами центра (x_0,y_0) и радиусом R и отрезок, заданный координатами своих концов (x_1,y_1) и (x_2,y_2).

Тогда для любой точки (x,y), принадлежащей окружности можно записать уравнение вида: $(x-x_0)^2 + (y-y_0)^2 = R^2$ (1)

Для отрезка удобно воспользоваться параметрическим уравнением прямой на плоскости, содержащей этот отрезок (параметр t):

$$\begin{cases} x = x_1(1-t) + x_2 t \\ y = y_1(1-t) + y_2 t \end{cases} \quad (2)$$

Подставив (2) в (1) получим:

$$(x_1(1-t) + x_2 t - x_0)^2 + (y_1(1-t) + y_2 t - y_0)^2 = R^2$$

Раскроем скобки и сгруппируем:

$$t^2(x_2^2 - 2x_1x_2 + x_1^2 + y_2^2 - 2y_1y_2 + y_1^2) +$$
$$t\big(2(x_1x_2 - x_1^2 - x_0x_2 + x_0x_1) + 2(y_1y_2 - y_1^2 - y_0y_2 + y_0y_1)\big) +$$
$$x_1^2 - 2x_1x_0 + x_0^2 + y_1^2 - 2y_0y_1 + y_0^2 - R^2 = 0$$

Получили квадратное уравнение относительно t. Дискриминант является показателем пересечения:

1) меньше 0 – у прямой и окружности нет общих точек (не пересекаются);
2) равен 0 – одна общая точка (касание прямой окружности);
3) больше нуля – две общие точки (точки пересечения прямой и окружности).

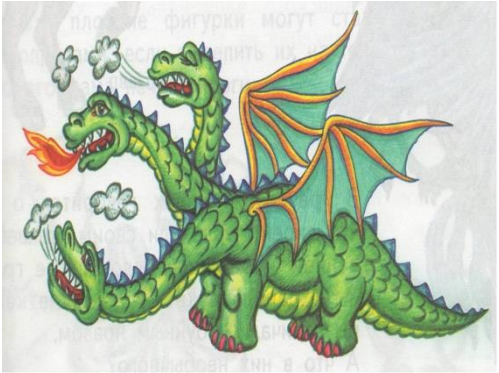
Учтём, что мы имеем дело с отрезком. Таким образом, решение может принять следующую форму:

```
//нахождение точек пересечения окружности и отрезка
//xp1,yp1,xp2,yp2 - координаты точек (одинаковы для касания)
//result - наличие пересечения (касания)
function CROSSING(x1,y1,x2,y2,x0,y0,r: real; var
xp1,yp1,xp2,yp2: real): boolean;
var a,b,c,d,k1,k2: real;
begin
//коэффициенты квадратного уравнения, полученного после решения
//системы уравнения окружности и
//параметрического уравнения прямой
  a := sqr(x2)-2*x2*x1+sqr(x1)+sqr(y2)-2*y2*y1+sqr(y1);
  b := 2*(x1*x2-sqr(x1)-x0*x2+x0*x1)+ 2*(y1*y2-sqr(y1)-
y0*y2+y0*y1);
  c := sqr(x1)-2*x1*x0+sqr(x0)+sqr(y1)-2*y1*y0+sqr(y0)-sqr(r);
  d:=b*b-4*a*c;
//Значение дискриминанта показывает наличие точек пересечения
if d < 0 then result := false
else
begin
  k1 := (-b+sqrt(d))/(2*a);
  k2 := (-b-sqrt(d))/(2*a);
```

```
  xp1 := x1*(1-k1)+x2*k1;
  yp1 := y1*(1-k1)+y2*k1;
  xp2 := x1*(1-k2)+x2*k2;
  yp2 := y1*(1-k2)+y2*k2;
//Проверка принадлежности точек пересечения отрезку
  if (min(x1,x2) <= min(xp1,xp2)) and
     (max(x1,x2) >= max(xp1,xp2)) and
     (min(y1,y2) <= min(yp1,yp2)) and
     (max(y1,y2) >= max(yp1,yp2))
  then result := true
  else result := false;
end;
end;
//---ОСНОВНАЯ ПРОГРАММА
var x1,y1,x2,y2,x0,y0,r,xt1,yt1,xt2,yt2: real;
    s: real = 0;
    n: integer;
BEGIN
   assign(input,'distance.in');
   reset(input);
   readln(x1,y1,x2,y2);
   readln(n);
   for var i := 1 to n do
   begin
     readln(x0,y0,r);
     if CROSSING(x1,y1,x2,y2,x0,y0,r,xt1,yt1,xt2,yt2)
     then s := s + sqrt(sqr(xt1-xt2)+sqr(yt1-yt2));
   end;
   close (input);
   assign(output,'distance.out');
   rewrite(output);
   writeln(s:0:2);
   Close (output);
END.
```

Задача 4. Число Змея Горыныча

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



Как известно всем жителям нашей страны с детства, трехглавый Змей Горыныч немало бед доставил Князю Киевскому. Вот и опять идет он войной на Киев-град. Послал князь за сильно-могучими 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 их не съест, даже на войну не отвлекается.

Поэтому князь кинул кличь:

– Всем людям земли Киевской собирать яблоки, и нести их ненасытному трехглавому Змею Горынычу!

Выстроился народ в ряд перед лежбищем чудища кровожадного. Вышел Змей Горыныч глянул на яблоки в ведрах и решил поиздеваться над людом трудолюбивым.

Заставил все яблоки, принесенные в ведрах, взвешивать (известно, что в ведро помещается до 9 кг яблок), а Змей вес (в кг) каждого ведра яблок записывал на листок и в результате у него длинню-ю-ю-ю-щее число получилось. Посмотрел на число Трехглавый Змей и сказал:

– Голов у меня 3, потому должно полученное число на 3 делиться, иначе головы мои обидеться могут. Вы, людишки, должны изменить количество яблок в одном из ведер, не меняя порядок следования ведер, таким образом, что если я вновь взвешу все ведра – у меня в записи «числа» поменяется только одна цифра, при этом новое число должно делиться на 3 и быть оно должно максимально возможным из всех таких чисел. Если справитесь с таким заданием, отпущу вас, а нет, так в яблочный салат в качестве приправы пойдете!!!!!!

От Вас **требуется** помочь люду киевскому, написав программу, которая выдает то число, что требует Змей Горыныч.

Формат входного файла

В первой строке записаны последовательно без пробелов веса ведер с яблоками x_i ($0 \le x_i \le 9$; $1 \le i \le 200$).

Формат выходного файла

В первой строке входного файла записано число, которое требует Змей Горыныч.

Пример входных и выходных данных

dragon.in	dragon.out
853	873

dragon.in	dragon.out
756	786

Решение

С учетом того, что в каждом ведре не более 9 кг яблок и ведра нельзя переставлять, решение задачи моделируется обработкой длинного числа, которое можно представить в виде строки или массива. Рассмотрим решение на примере обработки строк.

Входную последовательность чисел нужно считать в строку и работать с ней, как со строкой. Сначала определим остаток от деления исходного числа на 3. Для этого нужно посчитать сумму цифр исходного

числа (то есть сумму отдельных символов строки), и взять остаток от деления суммы на 3.

Если остаток от деления на 3 суммы цифр равен 0, то нужно увеличить какую-то цифру числа на 3, 6 или 9, так как в этом случае число уже делится на 3 и нужно изменить одну цифру так, чтобы остаток от деление на 3 не изменился.

Если остаток равен 1, то одну цифру числа нужно увеличить на 2, 5 или 8.

Если же остаток равен 3, то одну цифру числа нужно увеличить на 1, 4 или 7.

Поскольку нам нужно сделать новое число максимально большим, то нужно увеличить как можно более левую цифру исходного числа. Будем перебирать все цифры исходного числа слева направо, пока не найдем первую цифру, которую можно увеличить на нужную величину, при этом цифру нужно увеличить как можно больше (после увеличения эта цифра станет равна 7, 8 или 9).

Наконец, возможна ситуация, когда ни одной цифры увеличить нельзя, такое возможно, например, когда число состоит из одних цифр 9, или для числа 888, или для числа 7878. В этом случае нужно уменьшить последнюю цифру числа так, чтобы число стало кратным 3. Например, число 7878 будет преобразовано в число 7875.

Таким образом, решение может принять следующую форму:

```
var n:string;s,i,inc,d:integer;
begin
  assign(input,'dragon.in');
  reset(input);
  assign(output,'dragon.out');
  reset(output);
  readln(n);
  s := 0;
  for i:=1 to length(n) do
  s:=s+strtoint(n[i]);
  inc:=3-(s mod 3);

  for i:=1 to length(n) do
  if strtoint(n[i])+inc<=9
  then
  begin
    d:=strtoint(n[i])+inc;
    while d + 3 <= 9  do d:=d + 3;
    n[i]:=inttostr(d)[1];
    writeln(n);
    exit;
  end;
  i:=length(n) ;
  if inc<>3
  then d:= strtoint(n[i])-3+inc
  else d:= strtoint(n[i])-3;
  n[i]:=inttostr(d)[1];
```

```
  writeln(n);
  close(input);
  close(output);
end.
```