2015 год

Всероссийская олимпиада школьников по информатике
Муниципальный этап
9-11 классы

«Фа́нтик (Первобытная сказка)» – мультипликационный фильм Ефима Гамбурга по сценарию Бориса Заходера, 1975 год.

Задача A. Недопалиндромы — 20 баллов

Ограничение по времени, сек.	2
Ограничение по памяти, мегабайт	1
Имя входного файла	`palind.in`
Имя выходного файла	`palind.out`

Страшный зверь первобытной сказки Штуша-Кутуша очень не любил числа. Они вызывали у него подозрение. Однако все животные первобытной сказки что-нибудь да знали о них. Поэтому Штуша-Кутуша решил придумать такие числа, о которых животные ничего не знали. Он назвал их недопалиндромами. Числом недопалиндромом он считает такое число, сумма цифр которого является палиндромом, причём само исходное число таковым не является. Например, таким недопалиндромом является число 137, так как его сумма цифр равна 11. Число 11 – палиндром, так как читается одинаково слева-направо и наоборот. Число 22 уже не будет недопалиндромом (несмотря на то, что сумма цифр – 4 – палиндром), так как само является палиндромом. Помогите Штуше-Кутуше найти такие числа.

Требуется написать программу, выводящую все недопалиндромы в указанном интервале.

Формат входного файла

В первой строке – два натуральных числа a, b ($1 \le a, b \le 32767$) – интервал чисел для поиска недопалиндромов.

Формат выходного файла

Выходной файл должен содержать N строк – числа недопалиндромы в интервале [a, b] (включая a и b), либо сообщение «NO», если таковых нет.

Пример

palind.in	palind.out
820 840	821 830

Решение

Для решения задачи потребуется два фрагмента алгоритма. Первый (представлен в цикле) – предназначен для нахождения суммы цифр в числе.

Второй (оформлен в виде функции pol) – определение факта принадлежности числа к палиндромам.
Таким образом, пример решения может принять следующую форму.

```
//Функция определения является ли число палиндромом
function pol(st: integer): boolean;
begin
 var s := IntToStr(st);
 result := true;
 for var i :=1 to s.Length div 2 do
  if s[i] <> s[s.Length-i+1] then begin
   result := false; break;
 end;
end;
//--- ОСНОВНАЯ ПРОГРАММА
begin
 var a,b,sm: integer;
 var n: integer = 0;
//Чтение исходных данных
 var f := OpenRead('palind.in');
 read(f,a,b); f.Close;
 f := OpenWrite('palind.out');
//Решение задачи. Перебор чисел от a до b
 for var i := a to b do begin
  sm := 0;
  if not pol(i) then begin
//Находим сумму цифр числа
   for var j := 1 to IntToStr(i).Length do
    sm += StrToInt(IntToStr(i)[j]);
   if pol(sm) then begin writeln(f,i); n += 1; end;
  end;
 end;
//Проверяем счётчик количества недопалиндромов
 if n = 0 then writeln(f,'NO');
 f.Close;
end.
```

Задача B. Фильтр 20 баллов

Ограничение по времени, сек.	2
Ограничение по памяти, мегабайт	1
Имя входного файла	`speech.in`
Имя выходного файла	`speech.out`

Штуша-Кутуша, будучи примитивным, первобытным, невоспитанным и грубым зверем, постоянно некрасиво выражался. Чтобы исправить ситуацию, животные первобытной истории придумали фильтровать его речь. Так как Штуша-Кутуша говорил очень много и витиевато, то был придуман эвристический алгоритм, позволяющий отсеять ряд его подозрительных фраз в форме отдельных предложений. Предложение считалась подозрительным, если в нём встречается не менее двух слов, состоящих из трёх букв.

Требуется написать программу, позволяющую отфильтровать все подозрительные предложения.

Формат входного файла

Первая строка входного файла содержит одно число N ($1 \le N \le 32767$) – количество предложений, высказанных Штушей-Кутушей.

Далее N строк, состоящих из букв латиницы и знаков препинания, – предложения Штуши-Кутуши. Слова в предложениях отделены не менее чем одним пробелом.

Формат выходного файла

Выходной файл должен содержать M ($0 \le M \le 32767$) строк – все предложения исходного файла в том же порядке, кроме подозрительных.

Примеры

speech.in
6 Grazhdane pervobytnoi skazki! Ura mne - Shtushe-Kutushe! Davaite poimaem Fantika Raz, dva, tri, chetyre, piat. Ia idu ego iskat. Teplee, teplee, teplee...
speech.out
Grazhdane pervobytnoi skazki! Davaite poimaem Fantika Teplee, teplee, teplee...

Решение

Для решения задачи воспользуемся функцией (danger) оценки факта принадлежности строки к классу опасных (в ней встречается не менее двух слов, состоящих из трёх букв).

Таким образом, пример решения может принять следующую форму.

```
//Функция определения опасности предложения
function danger(st: string): boolean;
begin
//определим множество сисмволов из которых могут состоять слова
 var lets: set of char := ['A'..'Z','a'..'z'];
 var wrd := ''; //переменная для выделенных слов
 var count := 0; //счётчик опасных слов
 result := false;
 st += ' '; //концевой пробел для выделения последнего слова
 for var i := 1 to st.Length do begin
//посимвольно накапливаем буквы, пока они принадлежат слову
  if st[i] in lets then wrd += st[i] else begin
   if wrd.Length = 3 then count += 1;
   wrd := '';
  end;
//примем решение об опасности предложения
  if count = 2 then begin
    result := true; break;
  end;
 end;
end;
//---ОСНОВНАЯ ПРОГРАММА
begin
 var n: integer;
//Чтение исходных данных
 var fin := OpenRead('speech.in'); readln(fin,n);
 var fout := OpenWrite('speech.out');
//Решение задачи
 var s: string;
 for var i := 1 to n do begin
  readln(fin,s);
  if not danger(s) then writeln(fout,s);
 end;
 fin.Close; fout.Close;
end.
```

Задача C. Уничтожение 20 баллов

Ограничение по времени, сек.	2
Ограничение по памяти, мегабайт	1
Имя входного файла	`destruction.in`
Имя выходного файла	`destruction.out`

Штуша-Кутуша, готовясь к поимке Фантика, играл в Терминатора. Он брал последовательность двуцветных кубиков и уничтожал одну из последовательностей одного цвета, соединяя две полученные части вместе. Если соединённые части имели одинаковый цвет и тем самым образовывали последовательность, то они при объединении так же самоуничтожались. И так далее.

Требуется написать программу, которая по известной последовательности кубиков и номеру одного из кубиков в последовательности, которую уничтожил Штуша-Кутуша, определяет, какая конфигурация кубиков будет получена в итоге процессов уничтожения и самоуничтожения.

Формат входного файла

В первой строке N чисел ($1 \le N \le 32767$) 1 или 0 – коды цветов с первого до N кубика.

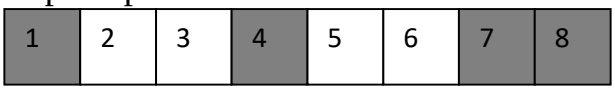
Во второй строке номер любого кубика последовательности, которую уничтожил Штуша-Кутуша в начале игры.

Формат выходного файла

Одна строка – полученная конфигурация кубиков после самоуничтожения или сообщение «NO» в случае, если уничтожены все кубики.

Примеры

Рисунок для первого примера:



destruction.in	destruction.out
10010011 6	100
11100111001 7	NO

Решение

Решение задачи сводится к моделированию процесса уничтожения кубиков. Таким образом, пример решения может принять следующую форму.

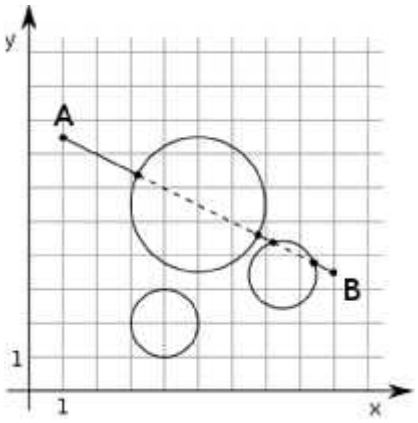
```
//Удаление последовательности кубиков одинакового цвета из s,
//начиная с n-го
function del_ident(s: string; n:integer ): string;
begin
 var a: char := s[n];
 while (a = s[n]) and (n > 1) do n -= 1;
//введём специальный символ «!», который указывает
//на текущий кубик
```

```
 if n = 1 then
   if s[1] = a then s[1] := '!' else
      begin s[2] := '!'; n := 2; end
  else
      begin s[n+1] := '!'; n += 1; end;
 n += 1;
 if n > s.Length then s[n-1] := '!' else begin
   while (s[n] = a) and (n < s.Length) do delete(s,n,1);
   if (n = s.Length) and (s[n] = a) then delete(s,n,1);
 end;
 result := s;
end;
//--- ОСНОВНАЯ ПРОГРАММА
 var k: integer; st: string;
begin
//Чтение исходных данных
 var f := OpenRead('destruction.in');
 readln(f,st,k);
 f.Close;
//Признак завершения процесса удаления
 var fin: boolean = false;
 repeat
 //удалим последовательность одинаковых
  st := del_ident(st,k);
 //найдём разделитель последовательностей после удаления
  k := pos('!',st);
 //если разделитель в начале или конце, то на этом всё
  if (k = 1) or (k = st.Length) then fin := true else begin
 //если по обе стороны разделителя разные кубики, то тоже всё
   if st[k-1] = st[k+1] then delete(st,k,1) else fin := true;
  end;
 until fin;
 //удалим уже ненужный разделитель
 delete(st,k,1);
 //вывод результата
 f := OpenWrite('destruction.out');
 if st.Length = 0 then write(f,'NO') else writeln(f,st);
 f.Close;
end.
```

Задача D. Ловушки — 20 баллов

Ограничение по времени, сек.	2
Ограничение по памяти, мегабайт	1
Имя входного файла	`jump.in`
Имя выходного файла	`jump.out`

Штуша-Кутуша охотился на Фантика. С этой целью он вырыл ряд круглых непересекающихся ям-ловушек. Однако Фантик был очень внимательным и спортивным слонёнком, поэтому двигаясь по прямолинейной траектории из точки A в точку B, подойдя к самому краю ямы-ловушки, он перепрыгивал её точно до следующего края, не отклоняясь от выбранной траектории.



Требуется написать программу, определяющую какое расстояние «напрыгал» (провёл в прыжке) Фантик с точностью до второго десятичного знака.

Формат входного файла

В первой строке вещественные числа x_A, y_A, x_B, y_B ($-10000 \le x_A, y_A, x_B, y_B \le 10000$) – координаты точек A и B.

Во второй строке число N ($1 \le N \le 32767$) количество ям-ловушек Штуши-Кутуши.

Далее N строк по 3 вещественных числа – x_k, y_k – координаты центра ямы-ловушки, R_k – её радиус ($-10000 \le x_k, y_k, R_k \le 10000$, $1 \le k \le N$).

Формат выходного файла

Одно число, с округлением до двух десятичных знаков, – «напрыганное» Фантиком расстояние.

Примеры

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

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:

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \quad (1)$$

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, группировка:

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

Получили квадратное уравнение относительно *t*. Дискриминант является показателем пересечения: меньше 0 – у прямой и окружности нет общих точек (не пересекаются), равен 0 – одна общая точка (касание прямой окружности), больше нуля – две общие точки (точки пересечения прямой и окружности). Учтём, что мы имеем дело с отрезком.

Таким образом, пример решения может принять следующую форму.

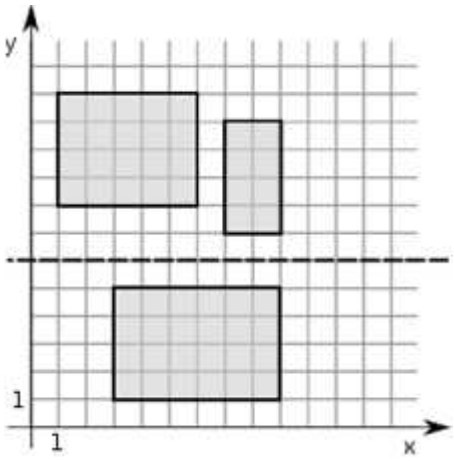
```
//нахождение точек пересечения окружности и отрезка
//xp1,yp1,xp2,yp2 - координаты точек (одинаковы для касания)
//result - наличие пересечения (касания)
function pnts(x1,y1,x2,y2,x0,y0,r: real; var xp1,yp1,xp2,yp2:
real): boolean;
 var a,b,c,d,t1,t2: real; intersect: boolean = true;
begin
//коэффициенты квадратного уравнения, полученного в результате
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);
//Значение дискриминанта показывает наличие точек пересечения
 intersect := (d >= 0);
 if not intersect then result := false else begin
   t1 := (-b+sqrt(d))/(2*a);
   t2 := (-b-sqrt(d))/(2*a);
   xp1 := x1*(1-t1)+x2*t1;
   yp1 := y1*(1-t1)+y2*t1;
   xp2 := x1*(1-t2)+x2*t2;
   yp2 := y1*(1-t2)+y2*t2;
//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
 var f := OpenRead('jump.in');
 readln(f,x1,y1,x2,y2);
 readln(f,n);
 for var i := 1 to n do begin
  readln(f,x0,y0,r);
  if pnts(x1,y1,x2,y2,x0,y0,r,xt1,yt1,xt2,yt2) then begin
   s += sqrt(sqr(xt1-xt2)+sqr(yt1-yt2));
 end; end;
 f.Close;
 f := OpenWrite('jump.out');
 writeln(f,s:0:2);
 f.Close;
END.
```

Задача E. Раздел 20 баллов

Ограничение по времени, сек.	3
Ограничение по памяти, мегабайт	1
Имя входного файла	`division.in`
Имя выходного файла	`division.out`

В Первобытном мире Штуша-Кутуша растил (однако плохо за ними ухаживал) колючки на полях, чтобы они превратилась в деревья и принесли волшебные плоды, исполняющие любое желание. Как только Фантик превратился в Элефанта, встал вопрос о более-менее справедливом дележе полей между Элефантом и Штушей-Кутушей (потому что Фантик очень хорошо ухаживал за колючками). Под более-менее справедливым дележом будем понимать такую ситуацию, когда суммарные площади доставшихся полей имеют минимальную разницу. После такого дележа необходимо построить забор, который будет отделять поля Элефанта от полей Штуши-Кутуши, причём так, чтобы он имел одинаковое расстояние до ближайшего поля любого оппонента, либо проходил ровно по их границе, не пересекая других полей. Все поля имеют прямоугольную форму с краями, параллельными осям координат. Для простоты было принято решение строить забор, также параллельный осям координат. В случае, когда можно построить как горизонтальный, так и вертикальный забор, герои отдают предпочтение горизонтальному. При этом если и тут есть несколько равных вариантов, то герои решили строить наиболее удалённый от начала системы координат забор.



Требуется написать программу, определяющую абсциссу (если вертикальный) или ординату (если горизонтальный) такого забора.

Формат входного файла

В первой строке одно целое число N ($1 \le N \le 5000$) – количество полей.

Далее N строк по 4 вещественных числа x_k, y_k, x_k, y_k ($-10000 \le x_k, y_k, x_k, y_k \le 10000$, $1 \le k \le N$) – координаты диагональных вершин прямоугольников-полей.

Формат выходного файла

Одно число, с округлением до двух десятичных знаков – абсцисса или ордината забора, либо сообщение «NO», если забор построить нельзя.

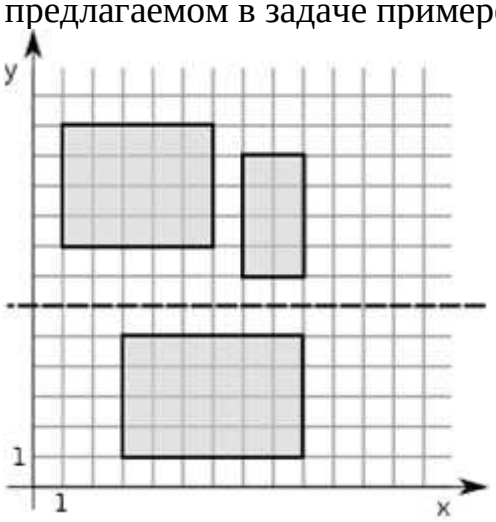
Примеры

division.in	division.out
3 1 8 6 12 3 1 9 5 7 7 9 11	6.00
4	NO

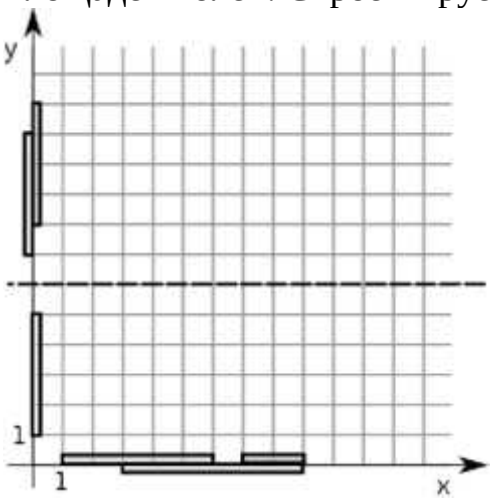
1 3 4 6 5 1 9 4 3 7 7 10 8 5 11 8	
3 3 3 1 1 6 6 3 4 7 9 9 7	6.50

Решение

Для решения этой задачи предпримем следующие действия. Рассмотрим на предлагаемом в задаче примере.



Начнём поиск для постройки вертикального забора. Для этого выполним следующие действия. Отсортируем все поля в порядке увеличения значения абсциссы левого края. Построим ряд значений суммарной площади в порядке увеличения абсциссы очередного поля: $S_0=s_0$, $S_1=S_0+s_1$, $S_2=S_1+s_2$ и т.д., где $s_{0..n-1}$ – значения площадей соответствующих полей, $S_{0..n-1}$ (в программе, обозначено a[k].s, где $k\in[0; n-1]$) – значения нарастающей общей суммы площадей полей. Спроектируем отсортированные поля на ось абсцисс.



Тогда, хорошо видно, что последовательный проход по массиву данных о полях с контролем проекций позволит нам выявить возможные координаты забора. В предложенном примере для вертикального забора их нет.

Если рассмотреть аналогичную ситуацию для горизонтального забора, соответственно проекции полей на ось ординат, то хорошо видно, что появился зазор, середина которого как раз является горизонтальным забором. Если таких ситуаций (как в примере для горизонтального забора) возникло несколько, следует выбрать ту, где разница между суммарными площадями минимальна и учесть все прочие дополнительные условия задачи.
Таким образом, пример решения может принять следующую форму.

```
 type cord = record x1,y1,x2,y2,s: real; end;
 var n: integer; x1,y1,x2,y2: real;
begin
//Чтение исходных данных
 var f := OpenRead('division.in'); readln(f,n);
//Воспользуемся динамическим массивом для хранения координат
 var a: array of cord; setlength(a,n);
 //перестановка координат (левое-нижнее, правое-верхнее)
 for var i := 0 to n-1 do begin
   readln(f,x1,y1,x2,y2);
   a[i].x1 := min(x1,x2); a[i].y1 := min(y1,y2);
   a[i].x2 := max(x1,x2); a[i].y2 := max(y1,y2);
 end;
 f.Close;
//решаем задачу по x
//сортируем по координате x (левой)
 var t: cord;
 for var i := 0 to n-2 do
  for var j := n-1 downto 1 do
   if a[j-1].x1 > a[j].x1 then
 begin
  t := a[j-1]; a[j-1] := a[j]; a[j] := t;
 end;
//запоминаем суммарную площадь с предыдущими
  a[0].s := (a[0].x2-a[0].x1)*(a[0].y2-a[0].y1);
  for var i := 1 to n-1 do
    a[i].s := a[i-1].s+(a[i].x2-a[i].x1)*(a[i].y2-a[i].y1);
//начинаем поиск промежутков в проекции полей на ось абсцисс
 var mindsx: real = a[n-1].s;  //разница в площадях
 var k: integer = 0;
 var k1: integer; var k2: integer;  //номера полей между
которыми забор
 for var i := 1 to n-1 do begin
  if (a[k].x2 > a[i].x1) and (a[k].x2 < a[i].x2) then k := i;
  if (a[k].x2 = a[i].x1) or (a[k].x2 < a[i].x1)then begin
    if mindsx >= abs((a[n-1].s-a[i-1].s)-a[i-1].s) then begin
       mindsx := abs((a[n-1].s-a[i-1].s)-a[i-1].s);
       k1 := k; k2 := i;
    end;
    k := i;
 end; end;
 //запомним результат по x, т.к. будем пересортировывать массив
 var rest := a[k1].x2+(a[k2].x1-a[k1].x2)/2;
```

```
//решаем задачу по у
//сортируем по координате у
 for var i := 0 to n-2 do
  for var j := n-1 downto 1 do
   if a[j-1].y1 > a[j].y1 then begin
    t := a[j-1]; a[j-1] := a[j]; a[j] := t;
 end;
//запоминаем суммарную площадь с предыдущими
  a[0].s := (a[0].x2-a[0].x1)*(a[0].y2-a[0].y1);
  for var i := 1 to n-1 do
    a[i].s := a[i-1].s+(a[i].x2-a[i].x1)*(a[i].y2-a[i].y1);
//начинаем поиск промежутков в проекции полей на ось ординат
 var mindsy: real = a[n-1].s;  //разница в площадях
 k := 0;
 for var i := 1 to n-1 do begin
  if (a[k].y2 > a[i].y1) and (a[k].y2 < a[i].y2) then k := i;
  if (a[k].y2 = a[i].y1) or (a[k].y2 < a[i].y1)then begin
    if mindsy >= abs((a[n-1].s-a[i-1].s)-a[i-1].s) then begin
       mindsy := abs((a[n-1].s-a[i-1].s)-a[i-1].s);
       k1 := k; k2 := i;
    end;
    k := i;
 end; end;

//приступаем к выбору лучшего ответа
 var res: string;
 if (mindsx = mindsy) and (mindsy = a[n-1].s) then res := 'NO'
else begin
   if mindsy <= mindsx then str(a[k1].y2+(a[k2].y1-
a[k1].y2)/2:0:2, res)
     else str(rest:0:2, res);
 end;
//выводим ответ в файл
 f := OpenWrite('division.out');
 write(f, res); f.Close;
end.
```