Голубева Полина Николаевна

Балл: 33

Задача №1

Условие

Учительница математики Марина Григорьевна попросила старшеклассников составить программу для тестирования младших учеников по геометрии, которая будет выдавать задачи на проверку принадлежности точки - прямой.

Требуется написать программу, которая будет контролировать правильность ответов учеников.

Входные данные: в первой строке уравнение вида ax+by+c=0, где a, b и c - целые числа от 0 до 1000, а знаками операции могут быть как "+", так и "-". Во второй строке - целые числа - координаты проверяемой точки.

Выходные данные: если точка лежит на прямой - слово "YES", иначе - слово "NO" и через пробел - целая часть расстояния от точки до прямой.

Пример

Исходные данные	Результат
1x+1y+0=0 1 -1	YES
2x-3y+1=0 2 0	NO 1

Исходный код

```
program n_1;
const c=[' ','+','-','=','x','y'];
b=['1','2','3','4','5','6','7','8','9'];
var s,e,f,f1:string;
x,y,i,j,p,L,k,g,p1,p2,sum:integer;
d:real;
a,u:array[1..2] of integer;
t:array[1..4] of integer;
begin
readln(e);
readln(f);
s:=' '+f+' ';
L:=length(s);
p:=1;
x:=0;
while p<L do begin
if (s[p]=' ') and (not(s[p+1]=' ') ) then begin
j:=p+1;
while not(s[j+1]=' ') do inc(j);
inc(x);
val(copy(s,p+1,j-p),k,g);
a[x]:=k;
p:=j;
end;
inc(p);
end;


s:=' '+e+' ';
L:=length(s);
p:=1;
x:=0;
while p<L do begin
if (s[p] in c) and (not(s[p+1] in c) ) then begin
j:=p+1;
while not(s[j+1] in c) do inc(j);
inc(x);
val(copy(s,p+1,j-p),k,g);
t[x]:=k;
p:=j;
end;
inc(p);
end;


u[1]:=a[1]*t[1];
```

```
u[2]:=a[2]*t[2];


p:=1;
sum:=0;
j:=0;
i:=1;
while p<length(e)-2 do begin
if e[p]='+' then begin
sum:=sum+u[i] ;
inc(i);
inc(j);
end;
if e[p]='-' then begin
sum:=sum-u[i];
inc(i);
inc(j);
end;
if (j=2) and (e[p]='+') then
sum:=sum+t[3];
if (j=2) and (e[p]='-') then
sum:=sum-t[3];
inc(p);
end;


if sum=t[4] then
writeln('YES')
else begin
write('NO ');
d:=(t[1]*a[1]+t[2]*a[2]+t[3])/(sqrt(t[1]*t[1]+t[2]*t[2]));
if d<0 then d:=-1*d;
writeln(d:0);
end;


end.
```

Задача №2

Условие

В строке записано истинное логическое выражение с тремя переменными a, b и c. Над переменными применяются 2 операции: эквивалентность и исключающее «или». Требуется восстановить значения таблицы истинности функции, соответствующей этому выражению.

Исключающее «или» - булева функция двух переменных, результат которой истинен тогда и только тогда, когда один аргумент истинен, а второй - ложен.

Эквивалентность - логическое выражение, которое является истинным тогда, когда оба простых логических выражения (левая и правая части) имеют одинаковую истинность.

В рамках данной задачи будем обозначать исключающее «или» знаком "^", а эквивалентность - знаком "=" без кавычек.

Входные данные: логическое выражение, состоящее из имён переменных a, b, c, знаков исключающего «или» и эквивалентности. Длина выражения не превышает 20 символов.

Выходные данные: 8 цифр 0 и 1, записанных неразрывно и означающих значения функции для каждой из комбинаций значений переменных:

0, 0, 0;

0, 0, 1;

0, 1, 0;

0, 1, 1;

1, 0, 0;

1, 0, 1;

1, 1, 0;

1, 1, 1.

Пример

Исходные данные	Результат
a^b^c	01101001

a^b=b^c	10100101

Примечание: операция исключающего «или» имеет более высокий приоритет по сравнению с эквивалентностью.

Исходный код

Задача №3

Условие

Дана запись двух больших целых чисел в шестнадцатеричной системе счисления и их произведения, при этом в записи допущена одна ошибка.

Написать программу, которая выявит допущенную ошибку и исправит её.

Входные данные: три строки, в каждой из которых записано по шестнадцатеричному числу. Количество знаков в каждом из чисел не превышает 100. Цифры от A до F записаны заглавными латинскими буквами.

Выходные данные: исправленная запись в том же формате.

Пример:

Исходные данные	Результат
10 A A2	10 A A0
1F 27 49A	1F 26 49A

Исходный код

```
program n_3;
uses sysutils;
var s,e,f:string;
a:array[1..3] of integer;
b:array [1..3] of string;
x,i,j,L,sum,n,k,p: integer;
d,d1:longint;

function cmp(s1, s2: string): integer;
var i, c: integer;
begin
  c:=0;
  for i:=1 to length(s1) do
  begin
    if s1[i] <> s2[i] then
      c := c+1;
  end;
  {writeln(s1);
  writeln(s2);
  writeln(c);
  writeln;}
  cmp := c;
end;


begin
   x:=1;
   sum:=0;
   j:=1;
   for i:=1 to 3 do begin
   readln(s);
   L:=length(s);
   b[i]:=s;
   for p:=L downto 1 do begin
   val(s[p],n,k);
   if k<>0 then
   begin
case s[p] of
'A','a': n:=10;
'B','b': n:=11;
'C','c': n:=12;
'D','d': n:=13;
'E','e': n:=14;
'F','f': n:=15;
end;
sum:=sum+n*x;
end
else
```

```
sum:=sum+n*x;
x:=x*16;
end;
a[j]:=sum;
inc(j);
x:=1;
sum:=0;
end;


d:=a[1]*a[2];
d1:=d;

k:=0;
f:='';
while d>0 do begin
k:=d mod 16;
if k>=10 then
begin
case k of
10: e:='A';
11: e:='B';
12: e:='C';
13: e:='D';
14: e:='E';
15: e:='F';
end;
f:=e+f;
end
else
begin
str(k,e);
f:=e+f;
end;
k:=0;
e:='';
d:=d div 16;
end;
if d1=0 then
f:='0';

if (cmp(f, b[3]) = 1) and (length(f)=length(b[3])) then
begin
   writeln(b[1]);
   writeln(b[2]);
   writeln(f);
end
else begin
   d:=a[3] div a[2];
   d1:=d;

   k:=0;
   f:='';
   while d>0 do begin
   k:=d mod 16;
   if k>=10 then
   begin
   case k of
   10: e:='A';
   11: e:='B';
   12: e:='C';
   13: e:='D';
   14: e:='E';
   15: e:='F';
   end;
   f:=e+f;
   end
   else
   begin
   str(k,e);
   f:=e+f;
   end;
   k:=0;
   e:='';
   d:=d div 16;
   end;
   if d1=0 then
   f:='0';
   if (cmp(f, b[1]) = 1) and (a[3] mod a[2] = 0) then
   begin
      writeln(f);
      writeln(b[2]);
      writeln(b[3]);
   end
   else
   begin
      d:=a[3] div a[1];
      d1:=d;

      k:=0;
      f:='';
      while d>0 do begin
      k:=d mod 16;
      if k>=10 then
```

```
      begin
      case k of
      10: e:='A';
      11: e:='B';
      12: e:='C';
      13: e:='D';
      14: e:='E';
      15: e:='F';
      end;
      f:=e+f;
      end
      else
      begin
      str(k,e);
      f:=e+f;
      end;
      k:=0;
      e:='';
      d:=d div 16;
      end;
      if d1=0 then
      f:='0';
      writeln(b[1]);
      writeln(f);
      writeln(b[3]);
   end
end;

end.
```

Задача №4

Условие

Пете нравится решать судоку, и он задумался, как можно изменить эту головоловку, чтобы она стала интереснее. После размышлений ему пришла идея сделать судоку в 16-ричной системе счисления, но он засомневался, получится ли её решать так же, как обычную.

Правила 16-ричных судоку, которые придумал Петя: дано поле 16x16, разделённое на квадраты 4x4. Допустимы все возможные 16-ричные цифры от 0 до F, при этом не должно быть повторов одной цифры в строках и столбцах поля, а также в отдельных квадратах.

Требуется написать программу, которая заполнит недостающие клетки в заданном 16-ричном судоку.

Входные данные: 16 строк по 16 символов с цифрами от 0 до F. Пустые поля обозначены символами "_" (знак подчёркивания). Количество пустых полей не превышает 30.

Выходные данные: решённое судоку - 16 строк по 16 символов с цифрами от 0 до F, такие, что все пустые поля заполнены пропущенными цифрами.

Пример:

Исходные данные	Результат

12_456_890ABCDEF	1234567890ABCDEF
567890ABCDEF1234	567890ABCDEF1234
90ABCDEF12345678	90ABCDEF12345678
CDEF123_567890AB	CDEF1234567890AB
3_567890A_CDE_12	34567890ABCDEF12
7890ABCDEF123456	7890ABCDEF123456
ABCDEF1234567890	ABCDEF1234567890
EF12_4567890ABCD	EF1234567890ABCD
678_0ABCDEF12345	67890ABCDEF12345
234__7890A__DEF1	234567890ABCDEF1
DEF1234567890_BC	DEF1234567890ABC
0ABC_EF123456789	0ABCDEF123456789
890ABCDEF123456_	890ABCDEF1234567
_567890ABCDEF1_3	4567890ABCDEF123
_1234567890ABCDE	F1234567890ABCDE
BCDEF123456789_A	BCDEF1234567890A

Примечание: гарантируется, что исходные данные корректны, судоку может быть решено и решение единственное.

Исходный код

```
program Hello; // Pascal [fpc 3.0.4]

const
    N = 16;
    NN = 256;
    blockN = 4;
    empty = '_';
    allvar: set of char = ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'];
    digits : array [1..16] of char = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');

type
 t_ch = set of char;
    t_puzzle = array [1..N, 1..N] of char;

var
    puzzle, tmp: t_puzzle;
    l, i, j, pos: integer;
    str,ts: WideString;


function getRowValues (puzzle:t_puzzle; rowIndex: integer): t_ch;
    var
        i: integer;
    begin
        getRowValues:=[];
        for i := 1 to N do
            getRowValues:= getRowValues + [puzzle[rowIndex, i]];
    end;

function getColumnValues (puzzle:t_puzzle; columnIndex: integer): t_ch;
    var
        i: integer;
    begin
        getColumnValues:=[];
        for i := 1 to N do
            getColumnValues := getColumnValues + [puzzle[i, columnIndex]];
    end;

function getBlockValues (puzzle:t_puzzle; rowIndex, columnIndex: integer): t_ch;
    var
        i, j, blockRowStart, blockColumnStart: integer;
    begin
        blockRowStart := blockN * ((rowIndex - 1) div blockN) + 1;
        blockColumnStart := blockN * ((columnIndex - 1) div blockN) + 1;

        getBlockValues:=[];
        for i := blockRowStart to  blockRowStart + blockN - 1 do
            for j := blockColumnStart to blockColumnStart + blockN - 1  do
                getBlockValues := getBlockValues + [puzzle[i, j]];
```

```
    end;


function get_nonposible_set (puzzle:t_puzzle; rowIndex,columnIndex: integer): t_ch;
    begin
        get_nonposible_set:=[];
        get_nonposible_set:= get_nonposible_set + getRowValues(puzzle, rowIndex);
        get_nonposible_set:= get_nonposible_set + getColumnValues(puzzle, columnIndex);
        get_nonposible_set:= get_nonposible_set + getBlockValues(puzzle, rowIndex, columnIndex);
    end;

function number (posible_set:t_ch):integer;
    var
        i,num:integer;
    begin
        for i:=1 to N do
            if digits[i] in posible_set then
                inc(num);
        number:=num;
    end;

function solve (var puzzle:t_puzzle): boolean;

    type t_variation = record
        i, j, num: integer;
        posible: t_ch;
    end;

    var
        i, j, k, c, jj: integer;
        tmp_puzzle: t_puzzle;
        variations: array [1..NN] of t_variation;
        posible_set, nonposible_set: t_ch;
        solved:boolean;
    begin
        tmp_puzzle := puzzle;
        k:=1;
        solve := true;

        for i := 1 to N do
            for j := 1 to N do
                if tmp_puzzle[i,j] = empty then
                begin

                    nonposible_set := get_nonposible_set(tmp_puzzle, i, j);
                    posible_set := allvar - nonposible_set;
                    if posible_set = [] then
                        exit (false);
                    variations[k].i := i;
                    variations[k].j := j;
                    variations[k].posible := posible_set;
                    variations[k].num := number(posible_set);
                    k:= k + 1;
                end;
        k:=k-1;

        for i:=1 to k do
            for j:=1 to k-i do
                if variations[j].num>variations[j+1].num then
                begin
                    c:=variations[j].num;
                    variations[j].num:=variations[j+1].num;
                    variations[j+1].num:=c;
                end;

        for i:=1 to k do
        begin

            if variations[i].num = 1 then
                for j := 1 to N do
                    begin
                        if digits[j] in variations[i].posible then
                        begin
                            tmp_puzzle[variations[i].i, variations[i].j]:= digits[j];

                        end;
                    end
            else
                begin
                    solved:=false;
                    for j := 1 to N do
                        if digits[j] in variations[i].posible then
                            begin
                                tmp_puzzle[variations[i].i, variations[i].j]:= digits[j];
                                if solve(tmp_puzzle) then
                                    solved:=true;
                                    break;
                            end;
                    if solved = false then
                        exit (false);
                end;
        end;
```

```
    {for i := 1 to N do
            for j := 1 to N do
                write(tmp_puzzle[i,j]);}
        puzzle := tmp_puzzle;
    end;

//Начало основной программы
begin
str := '';
for i:=1 to 16 do begin
readln(ts);
str := concat(str, ts);
end;
l:= length(str);

    for i := 1 to N do
        for j := 1 to N do
        begin
            pos:=(i-1)*N + j;
            if (pos <= l) then
                puzzle[i,j]:= str[pos];
        end;

    if solve(puzzle) then
        for i := 1 to N do begin
        {writeln(i);}
            for j := 1 to N do
                write(puzzle[i,j]);
        writeln;
        end

    else
        write('No solusion');
end.
```

Задача №5

Условие

Двоичным деревом называется иерархическая структура данных, в которой каждый узел имеет не более двух потомков.

Двоичное дерево поиска - разновидность двоичного дерева, у которого оба поддерева (левое и правое) являются двоичными деревьями поиска; все узлы левого поддерева произвольного узла A меньше, чем значение самого узла A; все узлы правого поддерева произвольного узла A больше либо равны значению узла A.

Высотой узла называется максимальная длина нисходящего пути от этого узла к самому нижнему узлу, называемому листом. Высота корневого узла равна **высоте** всего **дерева**.

Дерево называется **сбалансированным**, если для любой его вершины высота левого и правого поддерева для этой вершины различаются не более чем на 1.

При **прямом обходе** дерева сначала обрабатывается ключ (значение) корня, затем - ключи левого и правого поддеревьев.

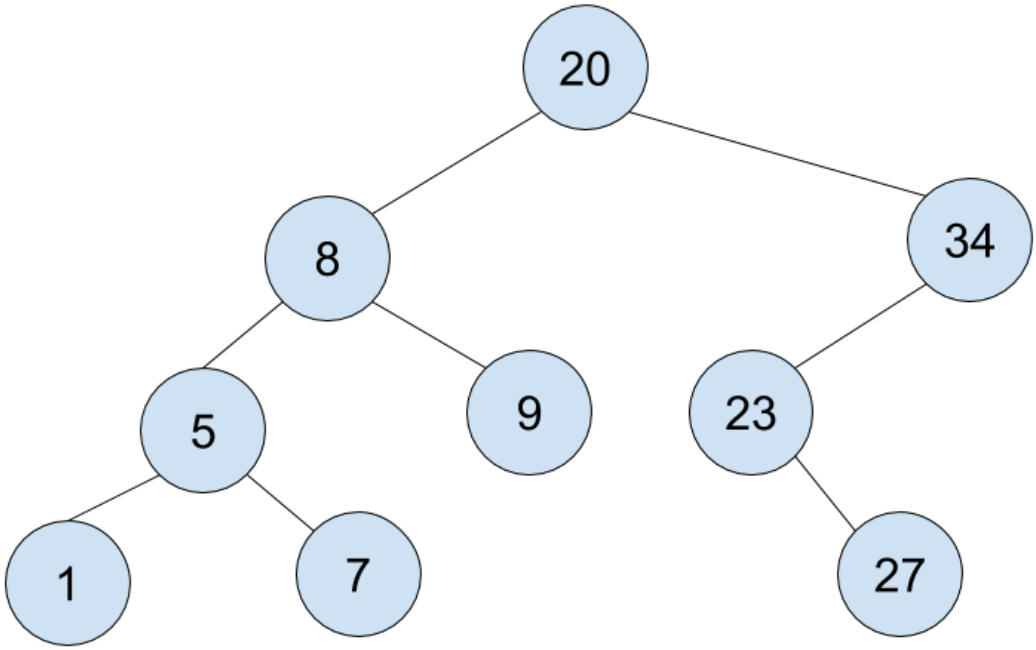
По заданным ключам вершин, полученным при прямом обходе, требуется определить, является ли дерево сбалансированным, а также определит высоту дерева.

Входные данные: ключи всех вершин двоичного дерева поиска в порядке прямого обхода. Каждый ключ - натуральное число, не превышающее 10_5. Каждое значение задано в отдельной строке. Последняя строка состоит из одного символа - точки.

Выходные данные: первая строка - слово YES, если дерево сбалансированное, и NO, если нет. Вторая строка - число, соответствующее высоте дерева.

Пример

Исходные данные и результат соответствуют дереву, изображённому на рисунке:



Исходные данные	Результат
20 8 5 1 7 9 34 23 27 .	NO 4

Исходный код