Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика".
2015 год. Заключительный этап. 10-11 классы. Билет 1. Задачи и решения.

Задача 1 (12 баллов). Пусть

$$S_n = \sum_{k=1}^{k=n} \left(\sqrt{1+\frac{k}{n^2}} - 1\right) u \lim_{n\to\infty} S_n \text{ существует.}$$

Найти предел S_n с точностью ε.
Входные данные. Стандартный входной поток содержит одно действительное число ε (0<ε<1).
Выходные данные. В стандартный выходной поток вывести одно действительное число - найденный предел.

Задача 2 (16 баллов). Используя уточнение корня уравнения по методу половинного деления, найти все действительные корни алгебраического уравнения $a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 = 0$ с точностью ε на отрезке $[-n, n]$.
Входные данные. Стандартный входной поток содержит семь действительных чисел a_0, a_1, a_2, a_3, a_4, a_5, ε и одно целое число n ($1 \le n \le 100$).
Выходные данные. В стандартный выходной поток вывести корни уравнения, если они есть, или слово NO, если их нет.

Примеры входных данных	Примеры выходных данных
1.0 2.5 -1.0 -3.5 -1.0 0.1 0.001 5	-2.379 -0.943 -0.470 0.079 1.213
1.0 1.0 1.0 -1.0 -1.0 0.1 0.001 5	-0.723 0.093 0.824
1.0 -1.0 1.0 -1.0 1.0 -1.0 0.001 2.5	1.000

Задача 3 (16 баллов). В некотором городе номера билетов для проезда в общественном транспорте кодируются в p-ичной позиционной системе счисления и состоят из $2k$ разрядов. Билет считается счастливым, если сумма первых k разрядов равна сумме последних k разрядов. Подсчитать количество счастливых билетов.
Входные данные. Стандартный входной поток содержит два целых числа: p ($2 \le p \le 16$) и k ($1 \le k \le 300$).
Выходные данные. В стандартный выходной поток вывести одно целое число - количество счастливых билетов.

Примеры входных данных	Примеры выходных данных
2 2	6
10 3	55252

Задача 4 (24 балла). На плоскости заданы N точек. Найти координаты центра окружности минимально возможного радиуса, внутри которой находятся все заданные точки.
Входные данные. Первая строка стандартного входного потока содержит целое число N ($3 \le N \le 1000$). В следующих N строках записаны пары действительных координат точек. Все координаты по модулю не больше 10^6.
Выходные данные. В стандартный выходной поток вывести три действительных числа – координаты центра окружности и радиус окружности. Результаты вывести с точностью 0.001.

Пример входных данных	Пример выходных данных
4 0 0 21 0 15 8 6 8	10.500 -1.625 10.625

Задача 5 (32 балла). Дан ориентированный связный граф, содержащий N вершин и M дуг. Вершины графа пронумерованы целыми числами от 1 до N. Вершины графа обмениваются между собой сообщениями. Время распространения сообщения по любой дуге равно 1 с. Вершина графа, получив сообщение, сразу отправляет его всем смежным вершинам по направлению дуги. Найти номера вершин, которые можно сделать «центральными», чтобы сообщение от них доходило до всех других вершин графа, а для наиболее удаленных вершин за минимальное время.
Входные данные. Первая строка стандартного входного потока содержит два целых числа: N ($1 \le N \le 100$) и M. В следующих M строках записаны номера вершин графа, соединенных дугой.
Выходные данные. В стандартный выходной поток вывести в порядке возрастания номера всех искомых вершин. Если ни одна вершина не подходит на роль «центральной», выведите 0.

Примеры входных данных	Примеры выходных данных
4 3 1 2 2 3 3 4	1
4 3 2 1 2 3 4 3	0

Решение задачи 1.

```
#include "stdafx.h"
#include <math.h>

double summa(int n)
{
	double result = 0;
	double t = (double)(n*n);
	for (int k = 1; k <= n; k++) result += sqrt(1.0 + k/t) - 1.0;
	return result;
}

int _tmain(int argc, _TCHAR* argv[])
{
	int n = 1;
	double s_new = sqrt(2.0) - 1.0, s_old, eps;
	scanf( "%lf", &eps );
	do
	{
		s_old = s_new;
		s_new = summa( ++n );
	} while ( abs(s_old - s_new) > eps );
	printf( "%lf\n", s_new );
	return 0;
}
```

Решение задачи 2.

```
#include "stdafx.h"
#include <math.h>

double a[6];

double f( double x )
{
	return a[0]*x*x*x*x*x + a[1]*x*x*x*x + a[2]*x*x*x + a[3]*x*x + a[4]*x + a[5];
}

double root(double x_left, double x_right, double eps)
{
	double x;
	do
	{
		x = (x_left + x_right)/2.0;
		if ( f( x )*f( x_left ) > 0 ) x_left = x;
		else x_right = x;
	} while ( abs( x_right - x_left ) > eps );
	return x;
}

int _tmain(int argc, _TCHAR* argv[])
{
	double x, eps, n, h = 0.1;
	for (int i = 0; i < 6; i++) scanf( "%lf", &a[i] );
	scanf( "%lf %lf", &eps, &n );
	x = -n;
	while (x < n)
	{
		if (f( x )*f( x+h ) < 0) printf( "%6.3lf\t", root( x, x+h, eps ) );
		x += h;
	}
	printf( "\n" );
	return 0;
}
```

Решение задачи 3.

```
#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
	unsigned int n[2][5001] = {0};
```

```
	int s, s_, k, k_, p;

	scanf( "%d %d", &p, &k );
	for ( s_ = 0; s_ < p; s_++ ) n[0][s_] = 1;
	for ( k_ = 2; k_ <= k; k_++ )
	{
		for ( s = 0; s <= k_*(p-1); s++ )
		{
			n[1][s] = 0;
			for ( s_ = 0; s_ < p; s_++ )
				if ( (s-s_ >= 0) && (s-s_ <= (k_-1)*(p-1)) )
					n[1][s] += n[0][s-s_];
		}
		for (int j = 0; j < 5001; j++) n[0][j] = n[1][j];
	}
	unsigned int n_total = 0;
	for ( s = 0; s <= k*(p-1); s++ ) n_total += n[0][s]*n[0][s];
	printf( "%d\n", n_total );
	return 0;
}
```

Решение задачи 5.

```
uses
  SysUtils, Math;
const
  MAX_SIZE = 100; INFINITY = 200;
type
  Matrix = array[1..MAX_SIZE, 1..MAX_SIZE] of byte;
var
  W, D: Matrix;
  i, j, k, u, v, n, m, maxrow, minmax: byte;
  A: array[1..MAX_SIZE] of byte;
  B: array[1..MAX_SIZE] of boolean;

begin
// Ввод исходных данных
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      W[i, j] := INFINITY;
  for i := 1 to MAX_SIZE do W[i, i] := 0;
  readln(n, m);
  for k := 1 to m do
  begin
    readln(u, v);
    W[u, v] := 1;
  end;
// Реализация алгоритма Floyd-Warshall
  for i := 1 to n do
    for j := 1 to n do
      D[i, j] := W[i, j];
  for k := 1 to n do
    for i := 1 to n do
      for j := 1 to n do
        D[i, j] := min(D[i, j], D[i, k] + D[k, j]);
// Определение строк матрицы D, в которых есть INFINITY
  for i := 1 to n do
  begin
    B[i] := false;
    for j := 1 to n do
      if D[i, j] = INFINITY then
      begin
        B[i] := true;
        break;
      end;
  end;

// Нахождение минимума среди максимумов строк матрицы D и запоминание максимумов
  minmax := INFINITY;
  for i := 1 to n do
  begin
    if B[i] then
    begin
      A[i] := INFINITY;
      continue;
```

```
    end
    else
    begin
      maxrow := 0;
      for j := 1 to n do
        if D[i, j] > maxrow then
          maxrow := D[i, j];
      A[i] := maxrow;
    end;
    if maxrow < minmax then minmax := maxrow;
  end;
// Подсчет числа строк матрицы D, содержащих INFINITY
  k := 0;
  for i := 1 to n do if B[i] then inc(k);
// Вывод результатов
  if k = n then writeln(0:4)
  else
    for i := 1 to n do if (not B[i]) and (A[i] = minmax) then write(i:4);
    writeln;
  readln;
end.
```

Олимпиада школьников «Шаг в будущее» по общеобразовательному предмету "Информатика".
2015 год. Заключительный этап. 10-11 классы. Билет 2. Задачи и решения.

Задача 1 (12 баллов). Пусть переменная x_n определяется следующим законом образования:

$$x_0 = \sqrt{a},\ x_1 = \sqrt{a+\sqrt{a}},\ x_2 = \sqrt{a+\sqrt{a+\sqrt{a}}},\ \ldots \text{ и } \lim_{n\to\infty} x_n \text{ существует.}$$

Найти предел x_n с точностью ε.
Входные данные. Стандартный входной поток содержит два действительных числа: a ($a > 0$) и ε ($0 < \varepsilon < 1$).
Выходные данные. В стандартный выходной поток вывести найденный предел.

Задача 2 (16 баллов). Используя уточнение корня уравнения по методу пропорциональных частей (методу хорд), найти все действительные корни алгебраического уравнения $a_0x^5 + a_1x^4 + a_2x^3 + a_3x^2 + a_4x + a_5 = 0$ с точностью ε на отрезке $[-n, n]$.
Входные данные. Стандартный входной поток содержит семь действительных чисел a_0, a_1, a_2, a_3, a_4, a_5, ε и одно целое число n ($1 \le n \le 100$).
Выходные данные. В стандартный выходной поток вывести корни уравнения, если они есть, или слово NO, если их нет.

Примеры входных данных	Примеры выходных данных
1.0 2.5 -1.0 -3.5 -1.0 0.1 0.001 5	-2.379 -0.943 -0.470 0.078 1.214
1.0 1.0 1.0 -1.0 -1.0 0.1 0.001 5	-0.723 0.093 0.824
1.0 -1.0 1.0 -1.0 1.0 -1.0 0.001 2.5	1.000

Задача 3 (16 баллов). Бросают N белых и M черных кубиков. На каждом выпадает число от 1 до 6. Считают суммы чисел, выпавших на белых и на черных кубиках, и полученные суммы перемножают. Найти количество разных способов получить в результате произведение P.
Входные данные. Стандартный входной поток содержит три целых числа: N, M ($1 \le N, M \le 250$) и P ($P > 0$).
Выходные данные. В стандартный выходной поток вывести одно целое число - количество способов.

Примеры входных данных	Примеры выходных данных
2 2 8	6
2 3 29	0

Задача 4 (24 балла). На плоскости дан простой многоугольник (т. е. без самокасаний и самопересечений, но не обязательно выпуклый). Проверить выпуклость многоугольника.
Входные данные. Стандартный входной поток содержит целое число N ($3 \le N \le 1000$) – количество вершин многоугольника и последовательность из N пар действительных координат вершин многоугольника. Все координаты по модулю не больше 10^6. Вершины многоугольника заданы в порядке их обхода против часовой стрелки.
Выходные данные. В стандартный выходной поток вывести слово YES, если многоугольник выпуклый, и слово NO в противном случае.

Примеры входных данных	Примеры выходных данных
3 0 0 3 0 0 3	YES
4 0 0 4 0 1 1 0 4	NO

Задача 5 (32 балла). Дан неориентированный связный граф, содержащий N вершин. Вершины графа пронумерованы целыми числами от 1 до N. Между любыми вершинами графа есть только один путь. Вершины графа обмениваются между собой сообщениями. Время распространения сообщения по любому ребру равно 1 с. Вершина графа, получив сообщение, сразу отправляет его всем смежным вершинам. Найти номера вершин, с которых может быть отправлено сообщение так, что максимальная задержка распространения сообщения была минимальной.
Входные данные. Первая строка стандартного входного потока содержит целое число N ($1 \le N \le 1000$). В следующих $(N - 1)$ строках записаны номера вершин графа, соединенных ребром.
Выходные данные. В стандартный выходной поток вывести номера всех искомых вершин.

Пример входных данных	Пример выходных данных
4 1 2 2 3 3 4	2 3

Решение задачи 1.

```
#include "stdafx.h"
#include <math.h>

int _tmain(int argc, _TCHAR* argv[])
{
	double a;
	double x_new, x_old, eps;
	scanf( "%lf %lf", &a, &eps );
	x_new = sqrt( a );
	do
	{
		x_old = x_new;
		x_new = sqrt( a + x_old );
	} while ( abs(x_old - x_new) > eps );
	printf( "%lf\n", x_new );
	return 0;
}
```

Решение задачи 2.

```
#include "stdafx.h"
#include <math.h>

double a[6];

double f( double x )
{
	return a[0]*x*x*x*x*x + a[1]*x*x*x*x + a[2]*x*x*x + a[3]*x*x + a[4]*x + a[5];
}

double root(double x_left, double x_right, double eps)
{
	double x, y, y_left, y_right;
	do
	{
		y_left = f( x_left );
		y_right = f( x_right );
		x = x_left - (y_left/(y_right - y_left)) * (x_right - x_left);
		y = f( x );
		if ( y_left*y > 0 ) x_left = x;
		else x_right = x;
	} while ( abs(y) > eps ); //abs( x_right - x_left ) > eps );
	return x;
}

int _tmain(int argc, _TCHAR* argv[])
{
	double x, eps, n, h = 0.1;
	for (int i = 0; i < 6; i++) scanf( "%lf", &a[i] );
	scanf( "%lf %lf", &eps, &n );
	x = -n;
	while (x < n)
	{
		if (f( x )*f( x+h ) < 0) printf( "%6.3lf\n", root( x, x+h, eps ) );
		x += h;
	}
	printf( "\n" );
	return 0;
}
```

Решение задачи 3.

```
uses
  SysUtils, Math;

const
  MAX_K = 250;
type
  Tval = extended;
var
  n1, n2, // основные таблицы количеств вариантов для белых и черных кубиков
  nprev, nthis: // вспомогательные таблицы
```

```
  array[-5..5*MAX_K] of Tval;
  k1, k2, // количество белых и черных кубиков
  p, // произведение
  temp, i, j, j1,
  mina1, maxa1, a1, a2: integer;
  res: Tval;

begin
  readln(k1, k2, p);
  if k1 > k2 then
  begin
    temp := k1;
    k1 := k2;
    k2 := temp;
  end;
  // первичная таблица, соответствующая одному кубику
  for i := -5 to 0 do nthis[i] := 0;
  for i := 1 to 6 do nthis[i] := 1;
  for i := 7 to 6*k2 do nthis[i] := 0;
  if k1 = 1 then n1 := nthis;
  // заполнение таблиц по формулам ...
  for i := 2 to k2 do
  begin
    nprev := nthis;
    for j := 1 to 6*i do
    begin
      nthis[j] := 0;
      for j1 := 1 to 6 do
        nthis[j] := nthis[j] ++ nprev[j - j1];
    end;
    if i = k1 then n1 := nthis;
  end;
  n2 := nthis;
  // границы для подходящих делителей числа p
  mina1 := max(k1, (p + 6*k2 - 1) div (6*k2));
  maxa1 := min(6*k1, p div k2);
  res := 0;
  // перебор возможных делителей числа p
  for a1 := mina1 to maxa1 do
    if p mod a1 = 0 then
    begin
      a2 := p div a1;
      res := res + n1[a1]*n2[a2];
    end;
  writeln(res:12:0);
  readln;
end.
```

Решение задачи 4.

```
#include "stdafx.h"

struct Point {
    double x, y;
};

int Direction(Point pi, Point pj, Point pk)
{
       int Result;
       double Temp = ((pk.x-pi.x)*(pj.y-pi.y)-(pj.x-pi.x)*(pk.y-pi.y));
       if (Temp < 0) Result = -1;
       else if (Temp > 0) Result = 1;
       else Result = 0;
       return Result;
}

int _tmain(int argc, _TCHAR* argv[])
{
       int N, k_pred, k_succ = 0;
       bool result = true;
       Point a[1000];
       scanf( "%d", &N );
       for (int i = 0; i < N; i++) scanf( "%lf %lf", &a[i].x, &a[i].y );
       k_pred = Direction(a[0], a[1], a[2]);
       for (int i = 1; i < N-2; i++)
```

```
	{
		k_succ = Direction(a[i], a[i+1], a[i+2]);
		if ( k_pred*k_succ < 0 )
		{
			result = false;
			break;
		}
		else k_pred = k_succ;
	}
	if (result )
	{
		k_succ = Direction(a[N-2], a[N-1], a[0]);
		if ( k_pred*k_succ < 0 ) result = false;
		else k_pred = k_succ;
	}
	if ( result )
	{
		k_succ = Direction(a[N-1], a[0], a[1]);
		if ( k_pred*k_succ < 0 ) result = false;
	}
	if ( result ) printf( "YES\n" );
	else printf( "NO\n" );
	return 0;
}
```

Решение задачи 5.

```
uses
  SysUtils, Math;
const
  MAX_SIZE = 1000; INFINITY = 2000;
type
  Matrix = array[1..MAX_SIZE, 1..MAX_SIZE] of word;
var
  W, D: Matrix;
  i, j, k, u, v, n, maxrow, minmax: word;
  A: array[1..MAX_SIZE] of word;
  flag: boolean;

begin
// Ввод исходных данных
  for i := 1 to MAX_SIZE do
    for j := 1 to MAX_SIZE do
      W[i, j] := INFINITY;
  for i := 1 to MAX_SIZE do W[i, i] := 0;
  readln(n);
  for k := 1 to n-1 do
  begin
    readln(u, v);
    W[u, v] := 1;
    W[v, u] := 1;
  end;
// Реализация алгоритма Floyd-Warshall
  for i := 1 to n do
    for j := 1 to n do
      D[i, j] := W[i, j];
  for k := 1 to n do
    for i := 1 to n do
      for j := 1 to n do
        D[i, j] := min(D[i, j], D[i, k] + D[k, j]);
// Определение недостижимых вершин в графе
  flag := false;
  for i := 1 to n do
    for j := 1 to n do
      if D[i, j] = INFINITY then
        flag := true;
// Нахождение минимума среди максимумов строк матрицы D и запоминание максимумов
  minmax := INFINITY;
  for i := 1 to n do
  begin
    maxrow := 0;
    for j := 1 to n do
      if D[i, j] > maxrow then maxrow := D[i, j];
    A[i] := maxrow;
    if maxrow < minmax then minmax := maxrow;
```

```
  end;

  if flag then writeln('ERROR')
  else
  begin
    for i := 1 to n do if A[i] = minmax then write(i:4);
    writeln;
  end;
  readln;
end.
```