9 Класс

Максимальная продолжительность – 210 мин.
Максимально возможное количество баллов – 500

Задача 1 (Время – 1 сек., память – 16 Мб, 100 баллов)

Сергей - начинающий программист, он посещает занятия по программированию в Центре «Точка роста». Занятия проходят еженедельно в один и тот же день недели. Сергей хочет написать программу, определяющую даты всех занятий, начиная с первого занятия и до конца года.

Помогите ему написать такую программу.

Входные данные: M - номер месяца (9, 10, 11, 12 – занятия начинаются в начале учебного года), ***D*** - день месяца, когда проходит первое занятие.

Выходные данные: даты всех занятий до конца календарного года (до 31 декабря включительно) в хронологическом порядке, по одной дате в строке, сначала месяц, затем день месяца, через пробел. Занятия проходят еженедельно, в тот же день недели, что и первое занятие. Каникулы отсутствуют.

Пример работы программы:

Ввод	Вывод
11 20	11 20 11 27 12 4 12 11 12 18 12 25

Решение:

В этой задаче нет больших чисел, и требуется всего лишь организовать правильный цикл по дням. Будем хранить в двух переменных m и d номер месяца и номер дня месяца. Затем в цикле будем увеличивать значение d на 7, что соответствует переходу на следующую неделю. Далее нужно аккуратно обработать переходы в следующий месяц. В сентябре 30 дней, поэтому если m = 9, а d > 30, то произошел переход в октябрь, поэтому нужно присвоить значение m := 10, а значение d уменьшить на 30. Аналогично, если m = 10, а d > 31, то произошел переход в ноябрь, поэтому нужно присвоить значение m := 11, а значение d уменьшить на 31. Если m = 11, а d > 30, то произошел переход в декабрь, нужно присвоить значение m := 12, а значение d уменьшить на 31. Наконец, если m = 12, а d > 31, то произошел переход в новый год и нужно завершить работу программы.

Пример решения задачи на алгоритмическом языке:

```
алг
нач
        цел m, d
        ввод m
        ввод d
        нц пока d <= 31
                вывод m, d, нс
                d :=d + 7
                если m = 9 и d > 30
                то
                        m := 10
                        d := d-30
```

```
            все
            если m = 10 и d > 31
            то
                    m := 11
                    d := d-31
            все
            если m = 11 и d > 30
            то
                    m := 12
                    d := d-30
            все
        кц
кон
```

Тесты для проверки:

ввод	вывод
11 20	11 20 11 27 12 4 12 11 12 18 12 25
12 29	12 29
12 3	12 3 12 10 12 17 12 24 12 31
11 15	11 15 11 22 11 29 12 6 12 13 12 20 12 27
11 11	11 11 11 18 11 25 12 2 12 9 12 16 12 23 12 30
10 26	10 26 11 2 11 9 11 16 11 23 11 30 12 7 12 14 12 21 12 28
10 16	10 16 10 23 10 30 11 6 11 13 11 20 11 27

ввод	вывод
	12 4 12 11 12 18 12 25
10 3	10 3 10 10 10 17 10 24 10 31 11 7 11 14 11 21 11 28 12 5 12 12 12 19 12 26
9 24	9 24 10 1 10 8 10 15 10 22 10 29 11 5 11 12 11 19 11 26 12 3 12 10 12 17 12 24 12 31
9 6	9 6 9 13 9 20 9 27 10 4 10 11 10 18 10 25 11 1 11 8 11 15 11 22 11 29 12 6 12 13 12 20 12 27
9 1	9 1 9 8 9 15 9 22 9 29 10 6 10 13 10 20 10 27 11 3 11 10 11 17 11 24 12 1 12 8

ввод	вывод
	12 15 12 22 12 29

За программу без ошибок начисляется 100 баллов, программа содержит несущественные ошибки, не влияющие на правильность решения – 50 баллов, в ином случае – 0 баллов.

Задача 2 (Время – 1 сек., память – 16 Мб, 100 баллов)

Циклическим сдвигом строки *s* называется строка $s_{k+1}s_{k+2}\dots s_n s_1\dots s_k$ для некоторого k $(0<=k<n)$, где n – длина строки s.

Для заданной строки требуется определить ее лексикографически минимальный циклический сдвиг, т.е. необходимо найти среди всех возможных циклических сдвигов строки тот, который идет первым в алфавитном порядке.

Входные данные: Строка, состоящая из заглавных букв английского алфавита. Длина строки от 1 до 1000 символов.

Выходные данные: Строка – минимальный лексикографический циклический сдвиг исходной строки.

Пример работы программы:

№	ввод	вывод
1	CAB	ABC
2	ABCAAAC	AAACABC

Решение:

В силу малых ограничений на длину строки мы можем перебрать все возможные сдвиги и выбрать наименьший среди них. Примеры решений:

```
//Pascal
var
   i,n : integer;
   s,m : string;
begin
   read(s);
   m := s;
   for i:=1 to length(s)-1 do begin
      s := copy(s,2,length(s)-1)+s[1];
      if s<m then m := s
   end;
   write(m)
end.
```

```
#Python
s = input()
print(min([s[i:]+s[:i] for i in range(len(s))]))
```

```
//C++
#include <bits/stdc++.h>
using namespace std;
int main(){
  int i;
```

```
  string s, m;

  cin >> s;
  m = s;
  for(i=1; i<s.length(); i++)
    s = s.substr(1)+s[0],
    m = min(m, s);

  cout << m;
  return 0;
}
```

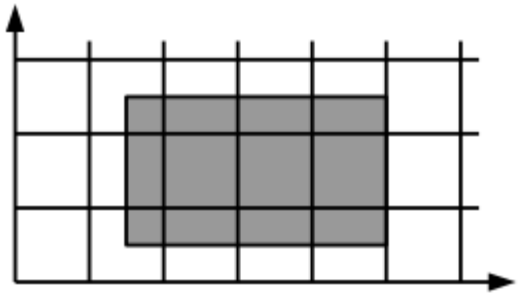
Программа работает верно – 100 баллов; программа содержит ошибки, но ход рассуждений верный – 20 баллов, в ином случае – 0 баллов.

Задача 3 (Время – 1 сек., память – 256 Мб, 100 баллов)

Стена покрыта квадратной плиткой со стороной M см. На стену повесили картину, известны координаты (X и Y) левого нижнего угла картины, её ширина и высота (W и H). Определите количество плиток, которые оказались частично или полностью закрыты картиной.

Входные данные: M — сторону плитки,. X и Y — координаты левого нижнего угла картины, W и H — ширина и высота картины. Ось OX направлена вправо, ось OY направлена вверх. Все числа целые, не превосходящие 2×10^9, числа M, W, H — положительные, числа X и Y — положительные или равны 0.

Выходные данные: количество плиток, полностью или частично закрытых картиной. Плитка считается закрытой картиной, если пересечение картины и плитки имеет ненулевую площадь, то есть касание картины и плитки не считается перекрытием.



Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов. Решение, правильно работающее только для случаев, когда все входные числа не превосходят 10^5, будет оцениваться в 70 баллов.

Решение:

Разобьём плоскость на плитки (квадраты со стороной M) и пронумеруем столбцы и строки квадратов начиная с 0. Сначала определим, в какой столбец плиток попадёт левая сторона картины, в какой столбец попадёт правая сторона картины, в какой ряд плиток попадёт нижняя сторона картины, в какой ряд плиток попадёт верхняя сторона картины. Для этого нужно координаты сторон плиток поделить на M. Левая сторона картины имеет координату X, правая — $X + W$, нижняя — Y, верхняя — $Y + H$. При этом при определении номера столбца левой стороны и номера строки нижней стороны нужно делить на M с округлением вниз, а при определении номера столбца правой стороны и номера строки верхней стороны нужно делить на M с округлением вверх. Затем определяем, количество столбцов и строк, которое занимает картина и перемножаем два полученных числа. Отметим, что в языках Pascal, C++, Java для получения полного балла

необходимо проводить вычисления с использованием 64-битных целых чисел, т. к. при использовании обычных целых чисел происходит переполнение целочисленной переменной при вычислении.

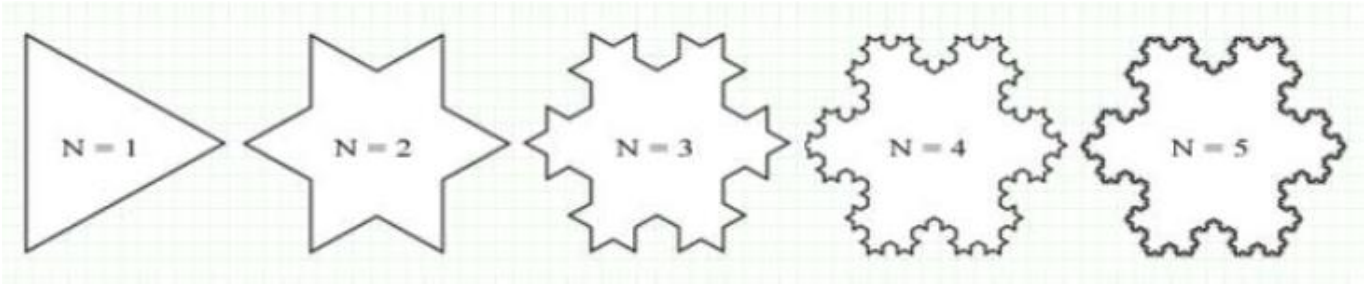
Пример решения:

Python
```
M = int(input())
X = int(input())
Y = int(input())
W = int(input())
H = int(input())
left = X // M
right = (X + W - 1) // M
bottom = Y // M
top = (Y + H - 1) // M
print((right - left + 1) * (top - bottom + 1))
```

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов. Решение, правильно работающее только для случаев, когда все входные числа не превосходят 10^5, будет оцениваться в 70 баллов. Правильное решение - 100 баллов

Задача 4 (Время – 1 сек., память – 16 Мб, 100 баллов)

Снежинка Коха – фрактальная кривая, которая строится на основе равностороннего треугольника, который представляет собой снежинку Коха порядка (N=1). Снежинка Коха К-го порядка строится из подобной кривой (К-1)-го порядка (К>1) путем замены каждой стороны данной фигуры четырьмя отрезками, каждый из которых представляет 1/3 от длины исходного отрезка (см. рисунок).



По заданному значению N требуется определить площадь фигуры, ограниченной снежинкой Коха N-го порядка, полагая, что при N=1 площадь равна единице.

Входные данные: натуральное число N ($N<=10^{18}$).

Выходные данные: площадь фигуры, ограниченной снежинкой Коха N-го порядка не менее, чем с шестью знаками после десятичной точки.

Пример работы программы:

№	Ввод	Вывод
1	1	1.000000
2	2	1.333333

Решения, работающие только для N<=5 будут оцениваться в 30 баллов.

Решения, работающие только для $N<=10^6$ будут оцениваться в 60 баллов.

Решение: Заметим, что изначально мы имеем равносторонний треугольник с площадью 1. При построении снежинки Коха 2-го порядка к нему добавляются еще 3 треугольника, площадь каждого из которых в 9 раз меньше площади исходного (т.к. линейные размеры уменьшены втрое). Так мы получили 12-сторонний многоугольник с равными длинами сторон.

Далее, при переходе от фигуры (i-1)-го порядка к i-му порядку очередное количество добавляемых треугольников увеличивается в 4 раза, а их площади уменьшаются в 9 раз.

Используем следующие обозначения:

P - площадь добавляемых треугольников,

K - количество добавляемых треугольников,

S - площадь всех треугольников, включая предыдущие шаги.

Таким образом, мы можем на каждом шаге вычислять количество добавляемых треугольников K, их площади P и добавлять к конечному ответу **S** их общую площадь **K·P**. В результате за N-1 шаг таких действий мы получим ответ. При этом следует учесть, что ответ нужно выводить с заданной точностью и что при N>50 в качестве ответа можно вывести площадь снежинки Коха 50-го порядка, т.к. последовательность площадей снежинки Коха сходится к числу 1.6 и при больших значениях N в качестве ответа можно выводить значение 1.6

Есть и другой способ решения этой задачи. Площадь снежинки Коха можно вычислить математически и представить решение одной формулой:

$$S_1 = 1$$

$$S_2 = S_1 + 3 \cdot \frac{1}{9} \cdot S_1 = 1 + \frac{1}{3}$$

$$S_3 = S_2 + 3 \cdot 4 \cdot \frac{1}{9^2} \cdot S_1 = 1 + \frac{1}{3} + \frac{1}{3} \cdot \frac{4}{9}$$

$$S_4 = S_3 + 3 \cdot 4^2 \cdot \frac{1}{9^3} \cdot S_1 = 1 + \frac{1}{3} + \frac{1}{3} \cdot \frac{4}{9} + \frac{1}{3} \cdot \frac{4^2}{9^2}$$

$$S_N = 1 + \frac{1}{3} + \frac{1}{3} \cdot \frac{4}{9} + \frac{1}{3} \cdot \frac{4^2}{9^2} + \ldots + \frac{1}{3} \cdot \frac{4^{N-2}}{9^{N-2}} = 1 + \frac{1}{3} \cdot \sum_{i=0}^{N-2} \left(\frac{4}{9}\right)^i = 1 + \frac{1}{3} \cdot \left(\frac{1 - \left(\frac{4}{9}\right)^{N-1}}{1 - \left(\frac{4}{9}\right)} \right) = 1 + \frac{1}{3} \cdot \left(1 - \left(\frac{4}{9}\right)^{N-1}\right)$$

Откуда легко видеть, что площадь бесконечной снежинки Коха составляет ровно 1.6, так как верно, что

$$\lim_{n \to \infty} \left(1 + \frac{3}{5} \cdot \left(1 - \left(\frac{4}{9}\right)^{N-1}\right) \right) = 1\frac{3}{5} = 1.6$$

Примеры программ:

```
//Pascal
var n : int64;
s,p,k : real;
begin
read(n);
if n>50 then n := 50;
s := 1;
p := 1;
k := 3;
while n>1 do begin
    p := p/9;
    s := s + k*p;
    k := k*4;
    dec(n)
end;
write(s)
end.
```

```
begin
write(1+3/5*(1-power(4/9, ReadReal-1)))
end
```

```
// C++
#include <bits/stdc++.h>
using namespace std;
int main(){
long long n;
double s=1,p=1,k=3;
cin >> n;
n = min(50LL, n);
while(--n)
   p /= 9,
   s += k*p,
   k *= 4;
cout << fixed << s;
return 0;
}
```

```
#include <bits/stdc++.h>
using namespace std;
int main(){
double n;
cin >> n;
cout << fixed << 1+0.6*(1-pow(4./9, n-1));
return 0;
}
```

```
#Python
print(1+0.6*(1-(4/9)**(int(input())-1)))
```

Решения, работающие только для N<=5, оцениваются в 30 баллов.
Решения, работающие только для $N<=10^6$, оцениваются в 60 баллов.
Решения, работающие при $N<=10^{18}$, оцениваются в 100 баллов.

Задача 5 (Время – 1 сек., память – 16 Мб, 100 баллов)

Напишите программу, решающую уравнение вида

$$\frac{a * x + b}{c * x + d} = v.$$

Числа *a,b,c,d,v* заданы, а *x* – неизвестно.

Входные данные: пять целых чисел *a,b,c,d,v,* не превосходящие 1000 по абсолютной величине.

Выходные данные: если уравнение не имеет решений, то выведите слово *NONE*. Если уравнение имеет ровно одно решение, то выведите строку вида *X=p/q*, где *p* - целое число, *q* – натуральное, *p* и *q* взаимно просты, а дробь является решением уравнения. Если уравнение имеет более одного решения, выведите слово *MULTIPLE*.

Пример работы программы:

№	ввод	вывод
1	1 2 3 4 5	X = -9/7
2	1 1 1 1 1	MULTIPLE
3	0 1 0 1 2	NONE

Решения, работающие только в случае существования единственного решения будут оцениваться в 50 баллов.

Решение:
Выполним несложные математические действия:

1) $\frac{a*x+b}{c*x+d} = v.$
2) $a * x + b = v * (c * x + d)$
3) $(a - v * c) * x = v * d - b$
4) $p = v * d - b \quad q = a - v * c$

5) $q * x = p$
6) $x = p/q$

Это решение верно только в том случае, когда уравнение имеет единственный корень. Однако, это не всегда так. Уравнение может не иметь корней или их может быть бесконечно много.

1) Когда c = 0 и d = 0 корней нет, т.к. знаменатель дроби исходного уравнения (1) равен нулю при любом x. Поскольку на ноль делить нельзя, то равенство не может быть верным при любом x.

2) Если в результате мы получаем, что p ≠ 0 и q = 0, то снова видим деление на ноль в (6). Заметим, что при этом не будет выполняться также (5). Т.е. в этом случае корней тоже нет.

3) Далее, если первые два пункта не выполнены и у нас p = 0 и q = 0, то мы получим бесконечное число решений, несмотря на неопределённость в (6). На самом деле это видно из (5): при любом x получаем, что 0 = 0. Однако, следует отметить, что при этом не любое значение x является решением. Об этом как раз следующий случай.

4) Если все предыдущие пункты не выполнены и мы получили определенную дробь в качестве ответа, то это не всегда возможный ответ. Дело в том, что при выполнении (2), (3), (5) и (6) может не выполняться (1). Это возможно только в том случае, когда при данном значении x знаменатель дроби обращается в ноль. Это произойдет, когда c×p+d×q = 0.

Во всех остальных случаях существует единственное решение, которое следует вывести в виде несократимой дроби. Для такого представления можно p и q разделить на НОД(p, q) и в случае отрицательного q поменять знаки у p и q. Для вычисления НОД можно использовать алгоритм Евклида, либо иной алгоритм.

Примеры решений:

```
//Pascal
var a, b, c, d, v, p, q, i : integer;
procedure Answer(ans : string);
begin write(ans); halt end;
begin
read(a, b, c, d, v);
p := v*d-b;
q := a-v*c;
if ((c=0) and (d=0)) or ((q=0) and (p<>0)) then Answer('NONE');
if (p=0) and (q=0) then Answer('MULTIPLE');
if c*p+d*q=0 then Answer('NONE');
if q<0 then begin p := -p; q := -q end;
p1:=abs(p);  q1:=abs(q);
while p1*q1 >0 do
    if p1>=q1 then p1:=p1 mod q1 else q1:= q1 mod p1;
write('X = ', p/(p1+q1), '/', q/(p1+q1));
end.

//C++
#include <bits/stdc++.h>
using namespace std;
int main(){
int a, b, c, d, v, p, q, gcd;
cin >> a >> b >> c >> d >> v;
p = v*d-b;
```

```
q = a-v*c;
if(c==0 && d==0 || q==0 && p!=0){cout << "NONE"; return 0;}
if(p==0 && q==0){cout << "MULTIPLE"; return 0;}
if(c*p+d*q==0){cout << "NONE"; return 0;}
if(q<0) p =- p, q =- q;
gcd = abs(__gcd(p,q));
cout << "X = " << p/gcd << "/" << q/gcd;
return 0;
}
```

```
#Python 3.5
from math import gcd
a, b, c, d, v = map(int, input().split())
p, q = v*d-b, a-v*c
if c==0 and d==0 or q==0 and p!=0:
print('NONE')
exit(0)
if p==0 and q==0:
print('MULTIPLE')
exit(0)
if c*p+d*q==0:
print('NONE')
exit(0)
if q<0:
p, q = -p, -q
print('X = ', p//gcd(p,q), '/', q//gcd(p,q), sep='')
```

Правильно работает программа и рассмотрены все случаи – 100 баллов.
Программа работает только для единственного решения – 50 баллов.
*Если программа не предусматривает анализ значений, перечисленных в пунктах 1-4 ((*c=0**) and (**d=0**); (**q=0**) and (**p<>0**); (**p=0**) and (**q=0**);* c*p+d*q=0 *) - -10 баллов за каждый пропущенный пункт.*