1. Кузнечик-2

Тема: разбор случаев, арифметика
Сложность: простая

Решение для одномерного случая рассмотрено в задаче 1 для 7-8 класса. Отдельно применяем его к для координаты X и Y и находим сумму.

Пример реализации:

```
var x,y,k:longint;
function jump(x:longint):longint;
var o:longint;
begin
  if x<0 then x:=-x;
  o:=x mod k;
  if o>k-o then
    o:=k-o+1;
  result:=x div k+o;
end;
begin
  read(x,y,k);
  writeln(jump(x)+jump(y));
end.
```

2. Без повторов

Тема: разбор случаев, работа со строками.
Сложность: простая

Чтобы строка из 0 и 1 не содержала двух одинаковых символов подряд, она должна иметь вид 010101… или 1010101010. Поэтому сравниваем количество изменений, которое нужно сделать для превращения строки один из двух вариантов и берем минимум.

Пример реализации:

```
var s:string;
  i,k1,k2:integer;
begin
  readln(s);
  k1:=0;
  k2:=0;
  for i:=1 to length(s) do
    if (ord(s[i])-ord('0'))<>(i mod 2) then inc(k1)
    else if (ord(s[i])-ord('0'))<>(1-i mod 2) then inc(k2);
  if k2<k1 then
    k1:=k2;
  writeln(k1);
end.
```

3. Перестановка

Тема: сортировка подсчетом, поиск минимума
Сложность: средняя

Так как в перестановке нет повторяющихся чисел, то нужно сначала убрать все повторяющиеся числа. Далее выбираем M, для которого количество добавляемых и удаляемых чисел минимально. В подзадаче 1 для этого можно перебрать все варианты M от 1 до 1000 и подсчитать сколько чисел больше M. Пусть O – количество различных чисел, U – количество чисел больше M среди них. Тогда для превращения в перестановку нам придется добавить M-(O-U) чисел и убрать U чисел, итого M-(O-U) +U=M-O+2*U действий. Выбрав M с минимальным количеством действий, удаляем числа больше M и добавляем несуществующие числа меньшие или равные M.

В подзадаче 2 можно заметить, что нет необходимости в числах с очень большими значениями, так как добавление несуществующих чисел до перестановки потребует большого количества действий, и проще удалить такое большое значение за 1 действие. Можно ограничиться диапазоном чисел от 1 до 200000 и использовать сортировку подсчетом. Это ограничение можно определить, рассмотрев два примера: для последовательности {6,7,8,9,10} лучше добавить 5 чисел 1,2,3,4,5, чем удалить их все и добавить число 1, а для последовательности {7,8,9,10,11} лучше удалить все числа и добавить число 1, чем добавлять 6 несуществующих чисел.

Пример реализации:

```
var c:array[1..200000] of boolean;
    n,m,k,km,i,x,o,u,ka:integer;
    a:array[1..200000] of integer;
begin
  read(n);
  o:=0; { количество оставшихся чисел }
  k:=0; { количество действий }
  for i:=1 to 200000 do
    c[i]:=false;
  for i:=1 to n do
  begin
    read(x);
    if (x>200000) or c[x] then
    begin { удалить число }
      inc(k);
      a[k]:=-x;
    end
    else
    begin { пока оставить число }
      inc(o);
      c[x]:=true;
    end;
  end;
  if not c[1] then { добавить 1 }
  begin
    inc(k);
    inc(o);
    a[k]:=1;
    c[1]:=true;
  end;
  m:=200000; { значение m для минимального количества действий }
  km:=200000-o; { минимальное количество действий }
  u:=0; { количество удаляемых элементов }
  for i:=200000 downto 1 do
  begin
    ka:=i-(o-u)+u; { количество действий для перестановки от 1 до I }
    if ka<km then
    begin { меньше — новый m и km }
      m:=i;
      km:=ka;
    end;
    if c[i] then
      inc(u);
  end;
  for i:=m+1 to 200000 do { удаляем все числа > m }
    if c[i] then
    begin
      inc(k);
      a[k]:=-i;
    end;
  for i:=1 to m do { добавляем несуществующие числа <= m }
    if not c[i] then
    begin
      inc(k);
      a[k]:=i;
    end;
  writeln(k); { печать действий }
  for i:=1 to k do
    if a[i]<0 then writeln(a[i])
    else writeln('+',a[i]);
end.
```

Пример реализации с использованием обычной сортировки:

```
import sys
n=int(input())
a=list(map(int,input().split()))
a.sort()
```

```
r=[] # список действий
u=[] # список уникальных чисел
p=-1
# убрать повторяющиеся
for x in a:
  if p!=x:
     p=x
     u.append(x)
  else:
     r.append(-x)
if u[0]!=1: # добавить 1, если её нет, так как M>=1
  u.insert(0,1)
  r.append(1)
l=len(u)
m=1
km=l-1
for i in range(l):
   ka=l-i+u[i]-i-2 # количество действий для получения перестановки M=u[i]
   if ka<km: # находим минимум действий и запоминаем m
     km=ka
     m=u[i]
p=1
for x in u:
  if x<=m:
    if p<x: # добавляем недостающие числа
      for i in range(p,x):
         r.append(i)
    p=x+1
  else: # удаляем числа >M
    r.append(-x)
print(len(r)) # печать действий
for x in r:
  if x<0:
    print(x)
  else:
    print('+'+str(x))
```

4. Идеальная стена-2

Тема: делители числа, разбор случаев
Сложность: выше средней

Подзадачи 1-4 совпадают с аналогичными подзадачами для 7-9 класса. В подзадаче 5 используются аналогичные рассуждения, но можно заметить, что для закрытия разрезов нужно (W-1)/(L-1) округленное вверх прямоугольных деталей (как минимум одним зубчиком нужно зацепиться за деталь из другого ряда, и только к последней детали справа можно ничто не прикреплять). Так A/B округленное вверх равно (A+B-1) div B (где div – деление нацело в языках программирования), то получаем формулу (W+L-3) div (L-1) для минимального количества прямоугольных деталей.

Пример реализации:

```
var a,b,s,w,h,L:int64;
    r1,r2:array[1..10000] of int64;
    i,n1,n2:integer;
function check(w,h:int64):boolean;
begin
  if h=1 then { высота стены 1 }
    result:=(w=L) and (b=1) or (w=1) and (b=0)
  else if w=1 then
    result:=b=0
  else if w=L then
    result:=b>0
  else if (w<L) or (b<(w+L-3) div (L-1))then
    result:=false
  else if w mod L=0 then
    result:=a >= L
  else
    result:=a>=h*(w mod L);
end;
```

```
begin
  read(a,b,l);
  s:=a+l*b;
  w:=1;
  n1:=0;
  n2:=0;
  while w*w<=s do
  begin
    if s mod w=0 then
    begin
       h:=s div w;
       if check(w, h) then
       begin
         inc(n1);
         r1[n1]:=w;
       end;
       if (w<>h) and check(h,w) then
       begin
         inc(n2);
         r2[n2]:=h;
       end;
    end;
    inc(w);
  end;
  writeln(n1+n2);
  for i:=1 to n1 do
    writeln(r1[i],' ',s div r1[i]);
  for i:=n2 downto 1 do
    writeln(r2[i],' ',s div r2[i]);
end.
```

5. Красивые числа

Тема: динамическое программирование
Сложность: выше средней

Для подзадачи 1 результат может быть найден путем полного перебора.
Пример реализации:

```
uses math;
var n,i,s,k,si:integer;
function sum(a:integer):integer; { сумма квадратов цифр }
var r,d:integer;
begin
  r:=0;
  while a>0 do
  begin
    d:=a mod 10;
    a:=a div 10;
    r:=r+d*d;
  end;
  sum:=r;
end;
begin
  read(n);
  k:=0;
  for i:=1 to n do
  begin
    s:=sum(i);
    si:=trunc(sqrt(s)); { проверка на полный квадрат }
    if si*si=s then
      inc(k);
  end;
  writeln(k);
end.
```

В подзадачах 2 и 3 нужно использовать таблицы для подсчета количества чисел с заданной суммой квадратов цифр. Расчет выполняется как в извествестной задаче о количестве счастливых билетов. В подзадаче 2 достаточно извлечь информацию из колонки, соответствующей степени k. В подзадаче 3

нужно сделать обратный проход по таблице для всех цифр.

Пример реализации:

```
#include <iostream>
#include <string>
using namespace std;
typedef long long ll;
ll sum2[20][1540];
int main()
{
  sum2[0][0]=1;
  for(int i=1;i<=19;++i)
    for(int s=0;s<=(i-1)*81;++s)
      for(int j=0;j<=9;++j)
      { sum2[i][s+j*j]+=sum2[i-1][s];
      }
  string n;
  cin>>n;
  ll k=0;
  for(int s=1;s*s<=n.size()*81;++s)
  { int s2=s*s;
    for(int i=0;i<n.size();++i)
    { int di=n[i]-'0';
      for(int d=0;d<di && s2-d*d>=0;++d)
        k+=sum2[n.size()-i-1][s2-d*d];
      s2-=di*di;
    }
    if(s2==0) ++k;
  }
  cout<<k<<"\n";
}
```