Разбор задач муниципального этапа олимпиады по информатике

1. Покраска забора (все классы)

Тема: вывод формулы, разбор случаев
Сложность: простая
Сначала нужно поменять границы, так чтобы первая граница диапазона была меньше или равно второй границе. Затем находим длину пересечения диапазонов и вычисляем результат как сумма длин диапазонов минус длина пересечения.
Пример реализации:

```
uses math;
var a,b,c,d,t:integer;
begin
  read(a,b,c,d);
  if a>b then
  begin
    t:=a;
    a:=b;
    b:=t;
  end;
  if c>d then
  begin
    t:=c;
    c:=d;
    d:=t;
  end;
  t:=min(b,d)-max(a,c)+1;
  if t<0 then t:=0;
  writeln(b-a+1+d-c+1-t);
end.
```

2. Числа-палидромы (все классы)

Тема: работа со строками, основы многоразрядной арифметики (сравнение чисел, увеличение на 1)
Сложность: средняя
Превращаем считанное число в палиндром, переписывая цифры в обратном порядке во вторую половину строки. Если число получилось меньше исходного — увеличиваем первую половину числа на 1 и снова превращаем его в палиндром.
Пример реализации:

```
var s,p:string;
procedure palin(var s:string); { превращение числа в палиндром }
var i,k:integer;
begin
  for i:=1 to length(s) div 2 do
    s[length(s)-i+1]:=s[i];
end;
procedure inchalf(var s:string); { увеличение первой половины числа на 1 }
var k:integer;
begin
  k:=(length(s)+1) div 2;
  while k>0 do
  begin
    if s[k]='9' then
    begin
      s[k]:='0';
      dec(k);
    end
    else
    begin
      inc(s[k]);
      exit;
```

```
    end;
  end;
  s:='1'+s;
end;
begin
  readln(s);
  p:=s;
  palin(p); { делаем палиндром }
  if p<=s then { новое число меньше или равно исходному? }
  begin
    inchalf(p); { увеличиваем на 1, начиная с середины }
    palin(p); { снова делаем палиндром }
  end;
  writeln(p);
end.
```

3. Алгоритм (7-9 классы)

Тема: реализация программы по схеме алгоритма
Сложность: простая

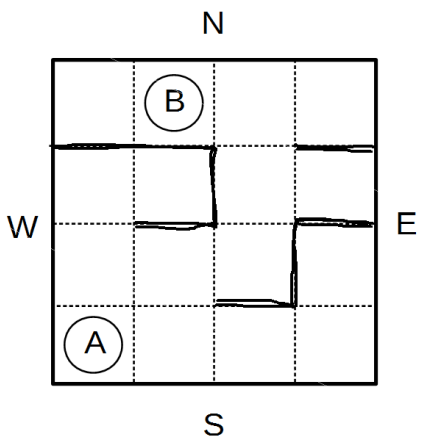
Пример реализации:

```
var i,n,k:longint;
begin
  k:=1;
  read(n);
  i:=2;
  while i*i<=n do
  begin
    while n mod i =0 do
    begin
      n:=n div i;
      if n mod i=0 then
      begin
        k:=k*i;
        n:=n div i;
      end;
    end;
    inc(i);
  end;
  writeln(k div 2);
end.
```

4. Робот в лабиринте (7-8 классы)

Тема: исполнители
Сложность: простая

Первая подзадача используется для проверки правильности восстановления схемы лабиринта. Она выглядит так:



Для проезда из A в B может быть 2 кратчайшие программы: NEENNW и ENENNW
Для решения второй подзадачи нужно предположить, что у нас на всех клетках стоит по роботу, и делать такие команды, которые постепенно уменьшают количество возможных позиций роботов до одной клетки, и привести всех их в клетку A.
Пример программы: EEEESWSSWWS

3. Медианный элемент (10-11 классы)

Тема: структуры данных
Сложность: сложная
Частичные решения: полный перебор для 1 подзадачи за $O(N^2)$
Кроме множества с порядковой статистикой или декартового дерева можно использовать деревья интервалов.

Пример реализации:

```
#include <ext/pb_ds/assoc_container.hpp>
#include <bits/stdc++.h>
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
typedef pair<int, int> pii;
typedef tree<pii, null_type, less<pii>, rb_tree_tag,
   tree_order_statistics_node_update> ordered_set;
int a[100000];

int main()
{ int n,k1,k2;
  ios::sync_with_stdio(0); // для ускорения ввода
  cin.tie(0);
  ordered_set q1,q2;
  cin>>n;
  for(int i=0;i<n;++i)
  {
     cin>>a[i];
     q2.insert(make_pair(a[i],i));
  }
  for(int i=0;i<n;++i)
  { q2.erase(make_pair(a[i],i));
    k2=q2.size()-q2.order_of_key(make_pair(a[i],-1));
    k1=q1.order_of_key(make_pair(a[i],n));
    if(k1>=k2)
    { cout<<(i+1)<<"\n";
      break;
    }
    q1.insert(make_pair(a[i],i));
  }
}
```

4. НОД подпоследовательностей (9-11 классы)

Тема: структуры данных, математика, свойства НОД
Сложность: сложная
Частичные решения: полный перебор для 1 и 2 подзадачи за $O(N^2 \log_2 10^{18})$ и новые НОД в подзадачах 1 и 3 можно отмечать флажками в массиве из 10^6 флажков вместо использования хэш-таблицы.

НОД подпоследовательности чисел не может возрастать при добавлении очередных элементов в подпоследовательность и таким образом не может принять более $\log_2 10^{18}$ значений (примерно 64).
Пусть $D_i = \{ НОД(a_j , a_{j+1}, \ldots , a_{i-1}, a_i) \mid j \le i \}$ при этом размер D_i не превышает $O(\log_2 a_i)$.
тогда можно рассчитать $D_{i+1} = \{ \gcd(v, a_{i+1}) \mid v \in D_i \} \cup \{ a_{i+1} \}$
Время работы алгоритма $O(N \log_2^2 10^{18})$

Пример реализации:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;

ll gcd(ll a, ll b)
{ return b?gcd(b,a%b):a;
}
int main() {
  ios::sync_with_stdio(0); // для ускорения ввода
  cin.tie(0);
  int N;
  cin >> N;
  ll a[64];
  unordered_set<ll> total;
  for (int ff = 0, nff = 0; cin >> a[ff++]; ff = nff, nff = 0)
      for (int i = 0; i < ff; ++i) {
            total.insert(a[nff] = gcd(a[i], a[ff-1]));
            if (!nff || a[nff] != a[nff-1]) ++nff;
      }
  cout << total.size() << endl;
}
```