Для олимпиады даются 4 задачи: 2 простых, средняя и сложная.
В следующей таблице дана сравнительная характеристика всех 4 задач

Задание	**Тематика**	**Сложность алгоритма**	**Сложность реализации**
1	Условия, циклы	Простая	Простая
2	Строки	Средняя	Простая
3	Динамическое программирование	Сложная	Средняя
4	Графы, множества, матрицы	Средняя	Сложная

Задача №1. Двоичный калькулятор. (100 баллов)

Можно вывести формулу с условиями, или просто применить цикл, т.к. ограничения очень небольшие. Далее приведено решение вторым способом, т.к. оно намного проще в реализации - нужно просто увеличивать текущее число на 3, если оно меньше конечного, и уменьшать на 2, если оно больше до тех пор, пока текущее не станет равным конечному. Очевидно, что в этом случае количество нажатий на кнопки калькулятора будет минимальным, т.к. мы никак не сможем улучшить такое решение.

```
var
  res, a, b : Integer;

begin
  read(a,b);
  res := 0;
  while a <> b do
  begin
    if a < b then
      inc(a, 3) else
      dec(a, 2);
    inc(res);
  end;
  writeln(res);
end.
```

Задача №2. Черно-белый пасьянс.

Легко заметить, что вариантов финального состояния карт всего 2 - или BWBWBW..., или WBWBWBWB...

Сравниваем количество отличий между заданной строкой и двумя финальными вариантами, и выбираем вариант с минимальным количеством отличий — это и будет ответ.

```
const
  BW = 'BW';
var
  i,r1,r2 : integer;
  s : string;

begin
  readln(s);
  r1 := 0;
  r2 := 0;
  for i := 1 to length(s) do
    if s[i] = BW[i mod 2 + 1] then inc(r1) else inc(r2);
  if r1 < r2 then writeln(r1) else writeln(r2);
end.
```

Задача №3. Бесконечная последовательность. (100 баллов)

Во-первых заметим, что для определения количества единиц в последовательности от L до R достаточно научиться определять кол-во единиц в последовательности длины n. Если f(n) = кол-во единиц в последовательности длиной n, то ответ на задачу:

f(R+1)-f(L)

Функцию f(n) можно определить рекуррентно следующим образом:

- находим максимальное $p=2^k$ такое, что $p \le n$

- очевидно, что количество единиц в последовательности длиной $p=2^k$ - [p/2]

- оставшаяся часть последовательности от p+1 до n представляет собой инвертированную исходную последовательность длиной (n-p), количество единиц в ней равно количеству нулей в исходной последовательности длиной (n-p), т.е. (n-p-f(n-p))

Таким образом:

f(n) = [p/2] + (n-p-f(n-p))

с базой

f(0)=f(1)=0

```
var
  l,r : integer;

function f(n : longint) : longint;
var p : longint;
begin
  if n < 2 then result := 0 else
  begin
    p := 1;
    while 2*p <= n do p := p * 2;
    result := p div 2 + (n-p-f(n-p));
  end;
end;
```

```
begin
  read(l,r);
  writeln(f(r+1) - f(l));
end.
```

Задача №4. Игра с закрытыми глазами. (100 баллов)

Для определения того, может ли Маша выиграть в игру, необходимо после каждого хода Васи определять все возможные положения фишки Маши. Если после любого хода Васи у фишки Маши нет возможных положений, значит Вася выиграл. Если же после последнего хода Васи у Маши все еще остается хотя бы одно возможное положение, значит победила Маша.

Таким образом нам надо поддерживать множество возможных позиций фишки Маши и обновлять его после каждого хода. Это можно реализовать модифицированным поиском в ширину.

При заданных ограничениях можно использовать также более простой алгоритм полного сканирования лабиринта (его сложность - O(mwh), где m - количество ходов Васи, w, h - размеры лабиринта), при котором на каждом шаге мы храним булевскую матрицу - для каждой позиции в лабиринте True, если Маша может быть в этой позиции на данном шаге, False - если не может. После каждого хода Васи формируется новая матрица возможных позиций путем полного прохода по всем клеткам лабиринта в текущей матрице, и для тех, значение которых True - заполнение вокруг них значением True в новой матрице. После завершения заполнения новой матрицы - новая становится текущей, а бывшая текущая очищается.

Если на любом этапе текущая матрица пуста (в ней нет значений True), значит выиграл Вася. Если же после всех ходов Васи матрица не пуста, то выиграла Маша. Эта реализация приведена ниже.

```
type
  PArray = array[1..55, 1..55] of boolean;

var
  x,y,x1,y1,x2,y2,i,j,k,h,w : integer;
  a : Array[1..55] of string;
  moves : string;
  pcnt : Integer;
  pp, p1, p2 : ^PArray;
  p1win : Boolean;

procedure go(x, y : integer);
begin
  if (x >= 1) and (x <= w) and (y >= 1) and (y <= h) then
  if (x <> x1) or (y <> y1) then
  if a[y, x] = '.' then
  if not p2[y, x] then
  begin
    p2[y, x] := true;
```

```
    inc(pcnt);
  end;
end;

begin
  readln(h, w);
  new(p1);
  new(p2);
  FillChar(p1^, sizeof(p1^), False);
  pcnt := 1;
  for i := 1 to h do
  begin
    readln(a[i]);
    for j := 1 to w do
      if a[i,j] = 'A' then
      begin
        x1 := j;
        y1 := i;
        a[i, j] := '.';
      end else
      if a[i,j] = 'B' then
      begin
        p1[i, j] := True;
        a[i, j] := '.';
      end;
  end;
  readln(moves);
  p1win := False;
  for i := 1 to length(moves) do
  begin
    case moves[i] of
    'L': dec(x1);
    'R': inc(x1);
    'U': dec(y1);
    'D': inc(y1);
    end;
    if p1[y1, x1] then
    begin
      p1[y1, x1] := False;
      dec(pcnt);
    end;
    if pcnt = 0 then
    begin
      p1win := True;
      break;
    end;
    pcnt := 0;
    FillChar(p2^, sizeof(p2^), false);
    for y := 1 to h do
      for x := 1 to w do
        if p1[y, x] then
        begin
          go(x + 1, y);
```

```
          go(x - 1, y);
          go(x, y + 1);
          go(x, y - 1);
        end;
    if pcnt = 0 then
    begin
      p1win := True;
      break;
    end;
    pp := p1;
    p1 := p2;
    p2 := pp;
  end;

  if p1win then writeln('NO') else writeln('YES');

  dispose(p2);
  dispose(p1);
end.
```