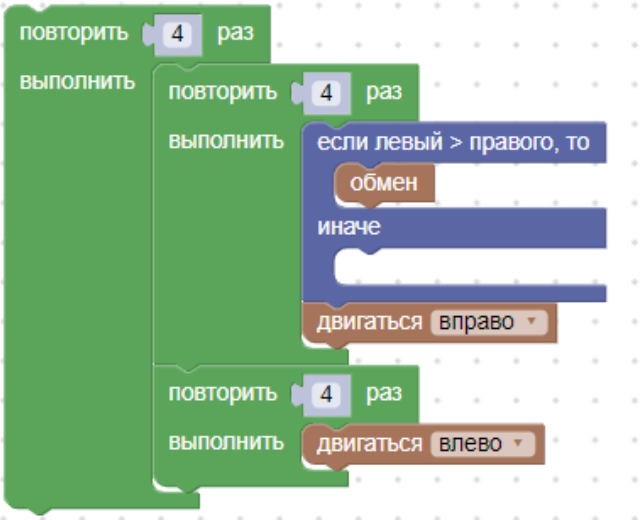
Разбор задач муниципального этапа олимпиады по информатике

1. 1,2,3,4,5 (7-8 классы)

Тема: исполнители
Сложность: простая

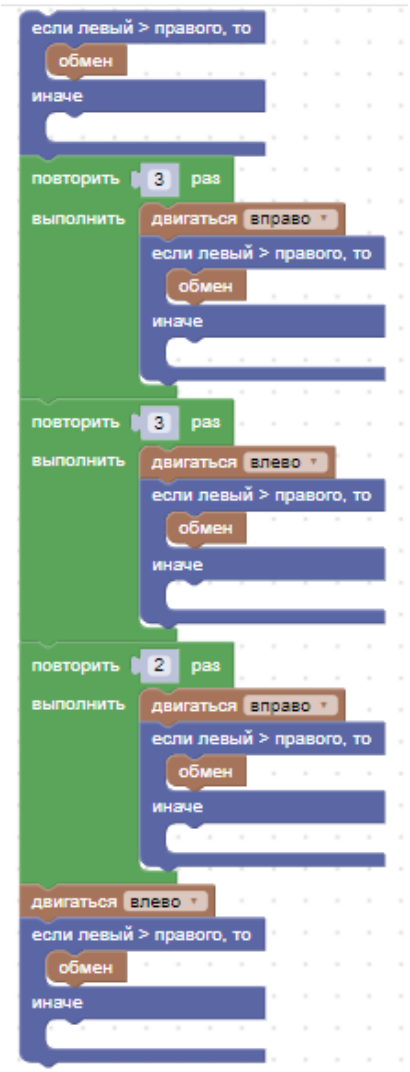
Решение подзадачи 2: Проходим по полю слева направо, и меняем числа, если порядок нарушен, затем возвращаемся на начальную позицию. После каждого прохода упорядоченность возрастает, сначала число 5 занимает конечную позицию и больше уже не перемещается, затем число 4, и так далее. После 4 повторений все числа окажутся на своих местах.

Пример реализации:



Решение подзадачи 3: Худшим случаем будет перестановка 5,4,3,2,1. Можно уменьшить количество перемещений, если при движении вправо перемещать большие числа, а при движении влево — меньшие. Итого получится 9 перемещений для упорядочения любой перестановки.

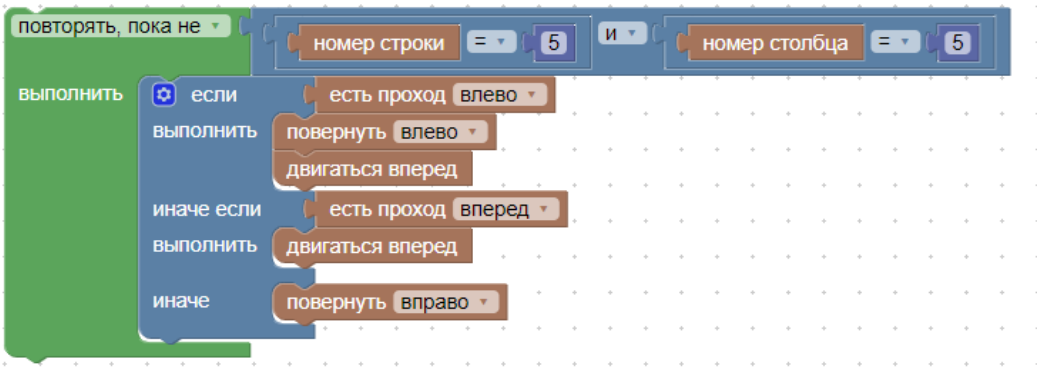
Пример реализации:



2. Лабиринт (7-8 класс)

Тема: исполнители
Сложность: простая

Необходимо реализовать указанный в задаче алгоритм. Пример реализации:



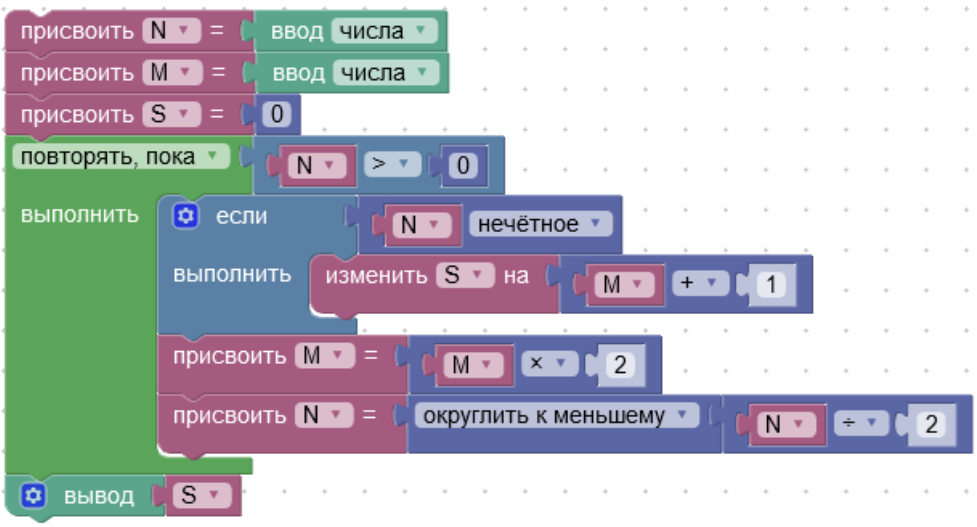
3 (1). Алгоритм (7-9 класс)

Тема: реализация программы по схеме алгоритма
Сложность: простая

Пример реализации:

```
var n,m,s:integer;
begin
  read(n,m);
  s:=0;
  while n>0 do
  begin
    if n mod 2 =1 then
      s:=s+m+1;
    n:=n div 2;
    m:=m*2;
  end;
  writeln(s);
end.
```

Пример реализации на Blockly:



4 (2,1). Смешанные команды (7-11 класс)

Тема: вывод математической формулы, разбор случаев
Сложность: ниже среднего

Подзадачу 1 можно решить с помощью цикла, выбирая 2 участников из группы с большим количеством людей, а 1 участника - из группы с меньшим количеством людей.

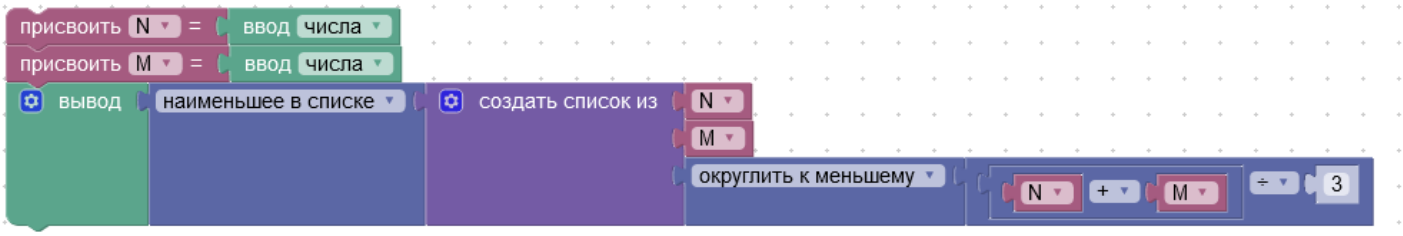
Полное решение задачи можно получить, разбирая основные случаи. Если N>M, то можно их поменять местами, количество команд от этого не изменится. Если N<M/2, то можно создать только N команд. Если разница не столь велика, можно создать M-N команд, выровняв количество людей в обеих группах, и затем можно создавать по 2 команды (1+2) и (2+1), когда в каждой группе останется менее 3 человек, в случае остатка 2 можно создать еще одну команду.

Пример реализации:

```
var n,m,k,t,d:int64;
begin
  read(n,m);
  if n>m then
  begin
    t:=n;
    n:=m;
    m:=t;
  end;
  if 2*n<=m then
    k:=n
  else
  begin
    d:=m-n;
    n:=n-d;
    m:=m-2*d;
    k:=d+2*(n div 3);
    n:=n mod 3;
    if n=2 then k:=k+1;
  end;
  writeln(k);
end.
```

Так как во втором случае после составления команд остаются не более двух человек, можно вывести целая_часть((N+M)/3).

Пример реализации на Blockly:



5 (3, 2). Форсаж (7-11 класс)

Тема: моделирование
Сложность: простая

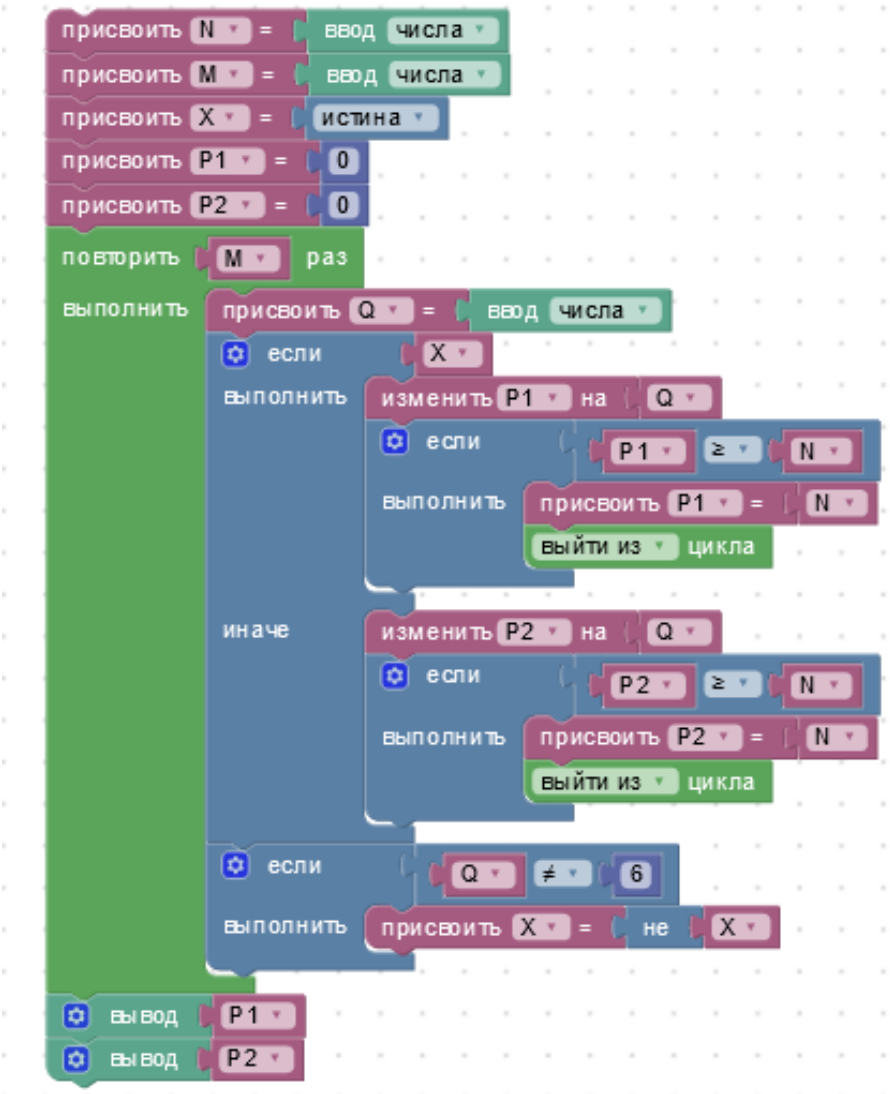
Выполняем моделирование по описанию в задаче.

Пример программы:

```
var n,m,p,i,q:integer;
    pos:array[1..2] of integer;
begin
   read(n);
   read(m);
   p:=1;
   pos[1]:=0;
   pos[2]:=0;
   for i:=1 to m do
   begin
     read(q);
     pos[p]:=pos[p]+q;
     if pos[p]>=n then
     begin
       pos[p]:=n;
       break;
     end;
     if q<>6 then p:=3-p;
```

```
   end;
   writeln(pos[1]);
   writeln(pos[2]);
end.
```

Пример реализации на Blockly:



4. Две строки (9-11 класс)

Тема: поиск максимума, полный перебор.
Сложность: ниже среднего

Перебираем все варианты для букв от A до Z и запоминаем лучший результат. Для максимизации количества повторяющихся букв используем правило: «Если есть нужная буква в одной из строк, то выбираем именно её, иначе можно выбрать любую».

Пример реализации:

```
var s1,s2:string;
    ch,maxch:char;
    i,kol,maxkol:integer;
begin
  readln(s1);
  readln(s2);
  maxch:='A';
  maxkol:=0;
  for ch:='A' to 'Z' do
  begin
    kol:=0;
    for i:=1 to length(s1) do
      if (ch=s1[i]) or (ch=s2[i]) then inc(kol);
    if kol>maxkol then
```

```
    begin
      maxkol:=kol;
      maxch:=ch;
    end;
  end;

  for i:=1 to length(s1) do
    if (maxch=s1[i]) or (maxch=s2[i]) then
      write(maxch)
    else
      write(s1[i]);
  writeln;
end.
```

5. Перестановка (9-11 класс)

Тема: динамическое программирование
Сложность: высокая

Используем динамическое программирование с параметрами DP[n][m][d][z], где

- n — текущий индекс в строке (1 <= n <= 1000),
- m — сколько нулей было использовано (0 <= m <= n).
- d — образовалась ли подстрока 00 (0 <= d <= 1),
- z — предыдущая цифра 0 (0 <= z <= 1).

Так как изменение цифры в строке происходит за счет обмена с противоположной цифрой, то можно изменение количества обменов учитывать только при смене 1 на 0.

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
const int inf=1e9;
const int maxn=1000;
int DP[maxn+1][maxn+1][2][2];
string str;
int main()
{
  cin>>str;
  int m=0, n=str.length();
  for(int i=0;i<n;++i)
    m+=str[i]=='0';
  for(int i=0;i<=n;++i)
    for(int j=0;j<=maxn; ++j)
      DP[i][j][0][0] = DP[i][j][1][0] = DP[i][j][0][1] = DP[i][j][1][1] = inf;
  DP[0][0][0][0]=0;
  for(int i=1;i<=n;++i)
  {
    // добавляем 1
    for(int j=0;j<=i; ++j)
    { DP[i][j][0][0] = min(DP[i-1][j][0][0],DP[i-1][j][0][1]);
      DP[i][j][1][0] = min(DP[i-1][j][1][0],DP[i-1][j][1][1]);
    }
    // добавляем 0
    for(int j=1;j<=i; ++j)
    { // нет 00
      DP[i][j][0][1] = (str[i-1]=='1') + DP[i-1][j-1][0][0];
      // был 00 или добавляем 0 к ...0
      DP[i][j][1][1] = (str[i-1]=='1') +
            min(DP[i-1][j-1][0][1], DP[i-1][j-1][1][0]);
    }
  }
  int r=min(DP[n][m][1][1],DP[n][m][1][0]);
  cout<<((r<inf)?r:-1)<<"\n";
}
```

3. Японский кроссворд (10-11 класс)

Тема: моделирование/идея, разбор случаев

Сложность: выше среднего

Для подзадачи 1 можно использовать моделирование, нарисовав этот рисунок и проверив, что числа соответствуют получившейся закраске.

```
type numbers=array[1..1000] of integer;
var pic:array[1..1000, 1..1000] of char;
    num:array[1..4] of numbers;
    i,j,n:integer;
procedure draw(var nn: numbers; si,sj,pi,pj,di,dj:integer);
var k,m,ti,tj:integer;
begin
  for m:=1 to n do
  begin
    ti:=si;
    tj:=sj;
    for k:=1 to nn[m] do
    begin
      pic[ti][tj]:='#';
      ti:=ti+di;
      tj:=tj+dj;
    end;
    si:=si+pi;
    sj:=sj+pj;
  end;
end;
function test(var nn:numbers; si,sj,pi,pj,di,dj:integer):boolean;
var m:integer;
begin
  for m:=1 to n do
  begin
    if nn[m]<>n then
      if pic[si+di*nn[m]][sj+dj*nn[m]]<>'.' then
      begin
        result:=false;
        exit;
      end;
    si:=si+pi;
    sj:=sj+pj;
  end;
  result:=true;
end;
begin
  read(n);
  for i:=1 to 4 do
    for j:=1 to n do
      read(num[i][j]);
  for i:=1 to n do
    for j:=1 to n do
      pic[i,j]:='.';
  draw(num[1],1,1,1,0,0,1);
  draw(num[2],1,n,1,0,0,-1);
  draw(num[3],1,1,0,1,1,0);
  draw(num[4],n,1,0,1,-1,0);
  if test(num[1],1,1,1,0,0,1) and test(num[2],1,n,1,0,0,-1) and
     test(num[3],1,1,0,1,1,0) and test(num[4],n,1,0,1,-1,0) then
    writeln('YES')
  else
    writeln('NO');
end.
```

Полное решение: проверяем для каждой полоски, которая не проходит через весь кроссворд, что закрывающая её белая клетка не попадает на полоски, проведенные с трех других сторон кроссворда.

```
var num:array[1..4, 1..100000] of integer;
    i,j,n:integer;
function isWhite(i,j:integer):boolean;
begin { ни одна из полосок закрашенных клеток не достает до клетки (i,j) }
```

```
  result := (num[1,i]<j) and (n-num[2,i]>=j) and (num[3,j]<i) and (n-num[4,j]>=i);
end;
begin
  read(n);
  for i:=1 to 4 do
    for j:=1 to n do
      read(num[i][j]);
  for j:=1 to n do
    if not( ((num[1,j]=n) or isWhite(j,num[1,j]+1)) and
            ((num[2,j]=n) or isWhite(j,n-num[2,j])) and
            ((num[3,j]=n) or isWhite(num[3,j]+1,j)) and
            ((num[4,j]=n) or isWhite(n-num[4,j],j))) then
    begin { одна из полосок достает до белой клетки,
            проверяем только для полосок < n }
      writeln('NO');
      halt(0);
    end;
  writeln('YES')
end.
```