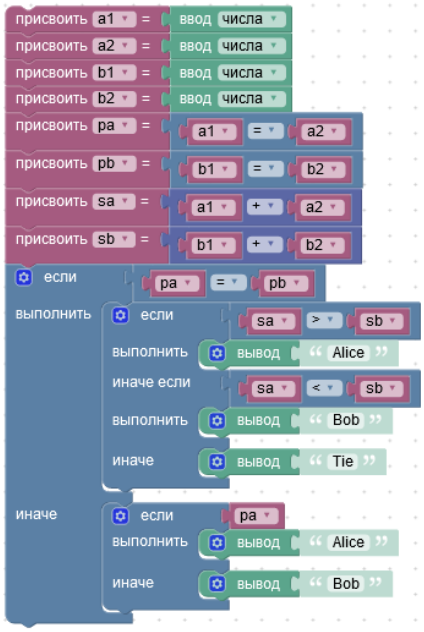
Разбор задач муниципального этапа олимпиады по информатике

1. Игра в кубики (все классы)

Тема: разбор случаев
Сложность: простая

Если у обоих игроков выпали пары или, наоборот, нет пар, то выводим игрока с большей суммой очков, иначе выводим игрока, у которого выпала пара

Пример реализации на Blockly:



Пример реализации:

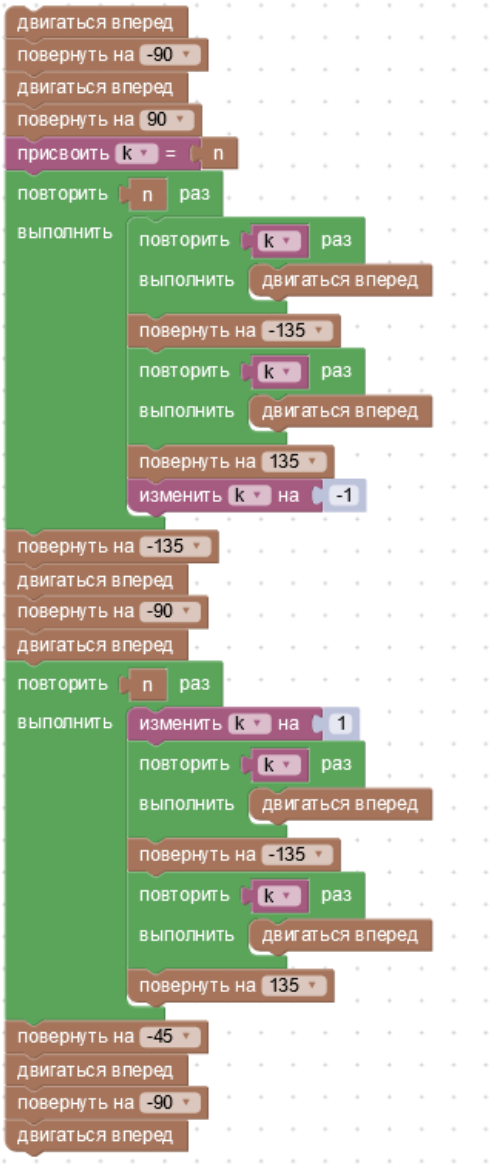
```
a1 = int(input())
a2 = int(input())
b1 = int(input())
b2 = int(input())
res_a = (a1 == a2, a1 + a2)
res_b = (b1 == b2, b1 + b2)
if res_a > res_b:
  print("Alice")
elif res_a < res_b:
  print("Bob")
else:
  print("Tie")
```

2. Ёлочка (7-8 классы)

Тема: исполнители, циклы
Сложность: простая

Рисуем зубцы с одной стороны с уменьшением размера, затем с другой, с увеличением размера.

Пример реализации:



3. Алгоритм (7-8 класс)

Тема: реализация программы по схеме алгоритма
Сложность: простая

Пример реализации:

```
n=int(input())
a=1
b=1
k=0
while a<=n:
  c=a+b
  b=a
  a=c
  k=k+1
while k>0:
  if b>n:
    print('0',end='')
  else:
    print('1',end='')
    n=n-b
  c=a-b
  a=b
  b=c
  k=k-1
```

```
var n,a,b,c,k:integer;
begin
  read(n);
  a:=1;
  b:=1;
  k:=0;
  while a<=n do
  begin
    c:=a+b;
    b:=a;
    a:=c;
    k:=k+1;
  end;
  while k>0 do
  begin
    if b>n then write('0')
    else begin write('1'); n:=n-b; end;
    c:=a-b;
    a:=b;
    b:=c;
    k:=k-1;
  end;
end.
```

4 или 2. Сумма цифр (все классы)

Тема: вывод формулы
Сложность: средняя

В подзадачах 1 и 2 (60 баллов) возможно решение грубой силой:

```
a = int(input())
b = int(input())
res = 0
for t in range(a, b + 1):
    n=t
    while n>0:
        res += n%10
        n//=10
print(res)
```

В подзадаче 3 нужно вывести формулы для суммы цифр чисел 1 до N. Для этого нужно рассмотреть каждый разряд отдельно: для k-го разряда можно выделить полные группы, в которых присутствуют все цифры от 0 до 9 (таких групп $[N/10^k]$, в каждой группе цифра повторяется 10^{k-1} раз, сумма цифр от 0 до 9 равна 45), затем неполные группы, в которых некоторая цифра присутствует 10^{k-1} раз (пусть x – это k-й разряд, тогда сумма цифр от 0 до x-1 равна x*(x-1)/2), и подсчитываем, сколько раз встречается x среди последних чисел: (N mod 10^{k-1} +1) раз. Чтобы найти результат в диапазоне нужно убрать из суммы цифр для чисел от 1 до B сумму цифр для чисел от 1 до A-1.

Пример реализации:

```
def sum(n):
  s=0
  d2=1
  for k in range(1,11):
    d1=d2
    d2*=10
    s+=(n//d2)*45*d1
    x=(n1//d1)%10
    s+=x*(x-1)//2*d1
    s+=x*(n%d1+1)
  return s
```

```
a=int(input())
b=int(input())
print(sum(b)-sum(a-1))
```

Пример реализации на Blockly

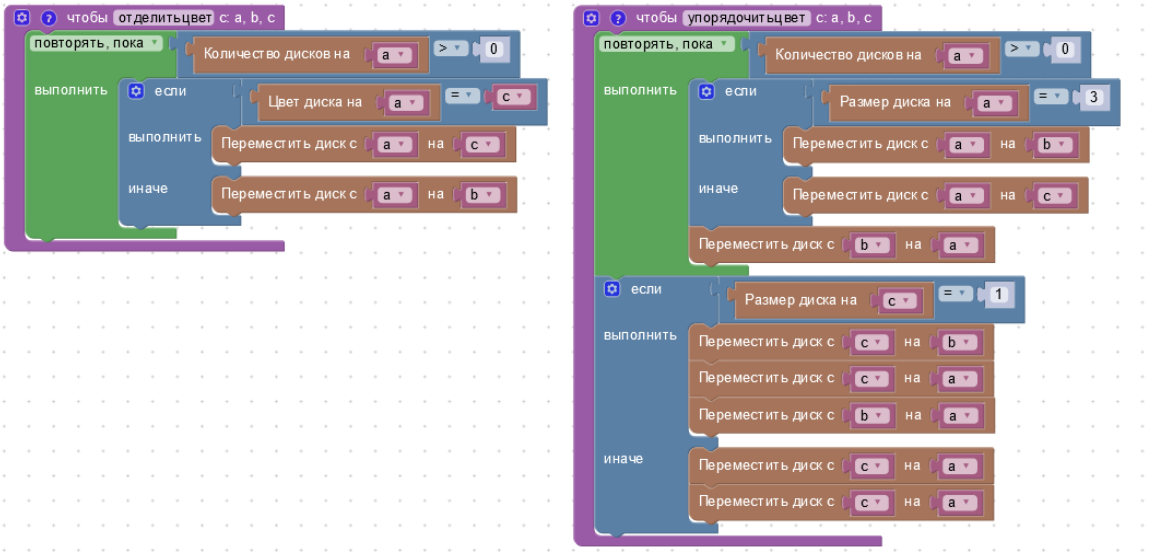
5. Пирамидки (7-8 класс)

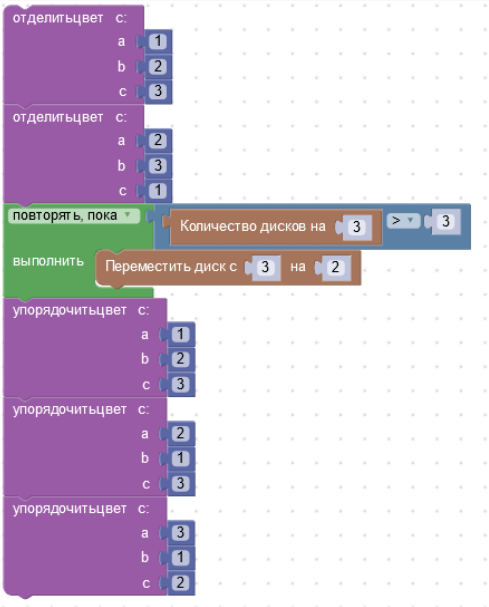
Тема: исполнители, подпрограммы, преобразование задачи к уже решенной
Сложность: ниже среднего

Сначала разделим диски по цветам. Для этого отделим диски желтого и красного цвета на столбик 2, а на столбик 3 поместим зеленые. После этого можно сделать разделение желтых и красных дисков, размещая красные на свободном столбике 1, но временно желтые складываем на столбике 3. Перемещаем желтые диски на свободный столбик 2.

Теперь нужно упорядочить диски на каждом столбике. В качестве временных можно использовать два других столбика. Это лучше сделать с

Пример реализации:





3. Колобок (9-11 класс)

Тема: перебор, оптимизация перебора с помощью геометрии.
Сложность: средняя

Для подзадач 1-5 можно сделать перебор вариантов расположения ошибки и моделирование движения робота. Если использовать рекурсивный перебор, то эти действия можно совместить.

Пример реализации:

```
import sys
```

```
sys.setrecursionlimit(2500)
s = input()
n = len(s)
k = int(input())

md=0
zamena='F+-'

def run(i,x,y,dx,dy,k):
  global md
  if i==n:
    if k==0:
      dist=abs(x)+abs(y)
      if md<dist:
        md=dist
    return
  ci=s[i]
  if k>0:
    for cz in zamena:
      if cz!=ci:
        if cz=='F':
          run(i+1,x+dx,y+dy,dx,dy,k-1)
        elif cz=='+':
          run(i+1,x,y,dy,-dx,k-1)
        else:
          run(i+1,x,y,-dy,dx,k-1)
  if ci=='F':
    run(i+1,x+dx,y+dy,dx,dy,k)
  elif ci=='+':
    run(i+1,x,y,dy,-dx,k);
  else:
    run(i+1,x,y,-dy,dx,k);

run(0,0,0,1,0,k)
print(md)
```

Для решения подзадачи 6 нужно оптимизировать движения робота между ошибками.

Пример реализации:

```
#include <iostream>
#include <string>
#include <cmath>
#include <algorithm>
using namespace std;
struct coord {
  int x,y;
  coord operator+(coord c2) const { return { x+c2.x, y+c2.y}; }
  coord operator-(coord c2) const { return { x-c2.x, y-c2.y}; }
  coord turn(char d) const { if(d=='+') return { y, -x}; else return { -y,x}; }
  int dist() const { return abs(x)+abs(y); }
};
char zamena[]="F+-";
int main()
{ int k;
  string prog;
  cin>>prog;
  cin>>k;
  vector<coord> c;
  c.push_back({0,0});
  int n=prog.size();
  c.resize(n+1);
  int d=0;
  coord dir[4]={{1,0},{0,1},{-1,0},{0,-1}};
  for(int i=0; i<n; i++)
  {
    if(prog[i]=='F')
      c[i+1]=c[i]+dir[d];
    else
```

```
    { c[i+1]=c[i];
      if(prog[i]=='+') d=(d+3)%4;
      else d=(d+1)%4;
    }
  }
  int md=0;
  if(k==0)
    md=c[n].dist();
  else if(k==1)
  { d=0;
    for(int i=0; i<n;++i)
    { coord c1=c[i]-c[0],c2=c[n]-c[i+1],c3;
      char ci=prog[i];
      for(int z=0;z<3;++z)
        if(zamena[z]!=ci)
        {
          if(zamena[z]=='-' && ci=='+' || zamena[z]=='+' && ci=='-')
c3=c1+c2.turn('-').turn('-');
          else if(zamena[z]=='-') c3=c1+c2.turn('-');
          else if(zamena[z]=='+') c3=c1+c2.turn('+');
          else if(zamena[z]=='F') c3=c1+dir[d]+c2.turn(ci=='+'?'-':'+');
//          cout<<i<<" "<<zamena[z]<<" "<<c3.x<<" "<<c3.y<<"="<<" "<<c1.x<<"
"<<c1.y<<" + "<<" "<<c2.x<<" "<<c2.y<<" "<<d<<"\n";
          md=max(md,c3.dist());
        }
      if(prog[i]=='+') d=(d+3)%4;
      else if(prog[i]=='-') d=(d+1)%4;
    }
  }
  else
  { d=0;
    for(int i=0; i<n-1;++i)
    { coord c1=c[i]-c[0];
      char ci=prog[i];
      for(int z=0;z<3;++z)
        if(zamena[z]!=ci)
        {
          int d1=d;
          if(zamena[z]=='+') d1=(d1+3)%4;
          else if(zamena[z]=='-') d1=(d1+1)%4;
          for(int j=i+1; j<n;++j)
          { coord c2=c[j]-c[i+1],c3=c[n]-c[j+1],c4,c5;
            char cj=prog[j];
            for(int y=0;y<3;++y)
              if(zamena[y]!=cj)
              {
                if(zamena[y]=='-' && cj=='+' || zamena[y]=='+' && cj=='-')
c4=c2+c3.turn('-').turn('-');
                else if(zamena[y]=='-') c4=c2+c3.turn('-');
                else if(zamena[y]=='+') c4=c2+c3.turn('+');
                else if(zamena[y]=='F') c4=c2+dir[d1]+c3.turn(cj=='+'?'-':'+');

                if(zamena[z]=='-' && ci=='+' || zamena[z]=='+' && ci=='-')
c5=c1+c4.turn('-').turn('-');
                else if(zamena[z]=='-') c5=c1+c4.turn('-');
                else if(zamena[z]=='+') c5=c1+c4.turn('+');
                else if(zamena[z]=='F') c5=c1+dir[d]+c4.turn(ci=='+'?'-':'+');
                md=max(md,c5.dist());
              }
            if(prog[j]=='+') d1=(d1+3)%4;
            else if(prog[j]=='-') d1=(d1+1)%4;
          }
        }
      if(prog[i]=='+') d=(d+3)%4;
      else if(prog[i]=='-') d=(d+1)%4;
    }
```

```
  }
  cout<<md<<"\n";
}
```

4. Сопротивление материалов (9-11 класс)

Тема: моделирование, разбор случаев.
Сложность: средняя

Для упрощения разделения нагрузки на левую и правую еще целые опоры сделаем Wi и Mi четными, умножив их на 2, а при выводе получившуюся нагрузку разделим на 2. Далее выполняем цикл для каждой опоры. Начальный вес нагрузки полагаем равным Mi-Wi, затем моделируем разрушение, по мере необходимости добавляя вес, если разрушение останавливается.

Пример реализации:

```
#include <iostream>
#include <vector>
using namespace std;
int main()
{ int n;
  cin>>n;
  vector<int> w(n);
  vector<int> m(n);
  for(auto &wi:w) { cin>>wi; wi*=2; }
  for(auto &mi:m) { cin>>mi; mi*=2; }
  for(int i=0;i<n;++i)
  {
    int d=m[i]-w[i];
    int r=w[i];
    int i1=i,i2=i; // начало и конец уже разрушенного участка
    while(i1>0 || i2<n-1)
    { if(i1>0 && i2<n-1)
      { // распределяем вес на 2 опоры
        int w1=w[i1-1]+(r+d)/2;
        int w2=w[i2+1]+(r+d)/2;
        if(w1>=m[i1-1])
        { i1--;
          r+=w[i1];
        }
        else if(w2>=m[i2+1])
        { i2++;
          r+=w[i2];
        }
        else
        { if(m[i1-1]-w1 < m[i2+1]-w2)
          { d+=2*(m[i1-1]-w1);
          }
          else
          { d+=2*(m[i2+1]-w2);
          }
        }
      }
      else if(i1==0)
      { // весь вес на i2
        int w2=w[i2+1]+(r+d);
        if(w2>=m[i2+1])
        { i2++;
          r+=w[i2];
        }
        else
        { d+=m[i2+1]-w2;
        }
      }
      else
      { // весь вес на i1
        int w1=w[i1-1]+(r+d);
        if(w1>=m[i1-1])
        { i1--;
```

```
          r+=w[i1];
        }
        else
        { d+=m[i1-1]-w1;
        }
      }
    }
    cout<<(d/2)<<" ";
  }
  cout<<"\n";
}
```

5. Эпоха империй (9-11 класс)

Тема: динамическое программирование
Сложность: выше среднего

Подзадача 1

Можно присоединить любую из территорий, но только одну за всю игру. Значит, можно выбрать территорию с максимальным размером.

Подзадача 2

Нельзя присоединять 2 территории подряд. То есть, требуется выбрать элементы с максимальной суммой при условии, что нельзя выбирать соседние элементы. Решим задачу методом динамического программирования. При движении по списку территорий можем вычислять две величины:

dp1[i] = максимальная территория, полученная в уже пройденной части списка.

dp2[i] = максимальная территория, полученная в уже пройденной части списка, если последняя текущая страна не присоединялась.

Подзадача 3

Стабильность тратится и не восстанавливается. Задача является классической задачей о рюкзаке: нужно выбрать территории с наибольшей суммарным размером ("предметы с максимальной ценностью"), не превысив суммарную величину штрафа ("вместимость рюкзака"). Решаем методом динамического программирования по параметрам (номер, текущая стабильность).

Подзадача 4

Аналогично подзадаче 3, но в формуле перехода учитываем скорость стабилизации P.

Пример реализации:

```
#include <vector>
#include <iostream>
using namespace std;
int main()
{
	int n, p;
	cin >> n >> p;
	vector<int> f(n), t(n);
	for (int i = 0; i < n; i++)
		cin >> f[i] >> t[i];

	vector<vector<int>> dp(n + 1, vector<int>(101, -1));
	dp[0][100] = 0;
	for (int i = 0; i < n; i++)
	{
		for (int j = 1; j <= 100; j++)
		{
			if (dp[i][j] == -1)
				continue;

			int nj = min(100, j + p);
			dp[i + 1][nj] = max(dp[i + 1][nj], dp[i][j]);

			nj = j - f[i];
			if (nj > 0)
				dp[i + 1][nj] = max(dp[i + 1][nj], dp[i][j] + t[i]);
		}
	}

	int best = 0;
	for (int t : dp[n])
```

```
            best = max(best, t);

        cout << best << endl;
        return 0;
}
```