A Ехаб и очередной конструктив

Можно решать перебором или формулой

Решение C++

```
#include <iostream>
using namespace std;
int main()
{
    int x;
    cin >> x;
    if (x==1)
      cout << -1;
    else
      cout << x-x%2 << ' ' << 2;
}
```

Решение Python

```
n=int(input())
if n==1:
    print(-1)
else:
    print(n,n)
```

B Тройки чисел

Первые два числа перебираем и смотрим сколько вариантов остается для третьего.

```
int a,b,c,s;
cin >> a >> b >> c >> s;


long long res=0;
for (int i=0;i<=a;i++)
for (int j=0;j<=b;j++)
{
      long long left  = s - i - j;
      left = min (left,(long long)c);
      if (left>=0) res+=left+1;
}

cout << res;
```

C Eevee

Задача на реализацию. Можно воспользоваться регулярными выражениями.

Решение C++

```
#include <iostream>
#include <algorithm>
#include <string>
#include <string.h>
using namespace std;

int main()
{
	int n; cin>>n;
	string s, P[]={"vaporeon", "jolteon", "flareon", "espeon", "umbreon",
"leafeon", "glaceon", "sylveon"};
	cin>>s;
	for(int i=0; i<8; i++)
	{
		if(P[i].size()==n)
		{
			for(int j=0; j<10; j++)
			{
				if(j==n-1) { cout<<P[i]<<endl; return 0; }
				if(s[j]=='.') continue;
				if(s[j]!=P[i][j]) break;
			}
		}
	}
}
```

Решение Python

```
n=int(input())
s=input()
l=['vaporeon','jolteon','flareon','espeon','umbreon','leafeon','glaceon','syl
veon']
import re
li=[i for i in l if len(i)==n]
for i in li:
    match=re.search(s,i)
    if match!=None:
        print(i)
        break
```

D Треугольники

Сначала докажем, что число неравных треугольников с вершинами в вершинах правильного n-угольника равно ближайшему к $n^2/12$ целому числу.

Решение

Пусть всего имеется N неравных треугольников с вершинами в вершинах правильного n-угольника, причем из них N_1 правильных, N_2 неправильных равнобедренных и N_3 разносторонних. Каждый правильный треугольник равен одному треугольнику с фиксированной вершиной A, неправильный равнобедренный — трем треугольникам с вершиной A, а разносторонний — шести. Так как всего имеется $(n - 1)(n - 2)/2$ треугольников с вершиной A, то $(n - 1)(n - 2)/2 = N_1 + 3N_2 + 6N_3$.
Ясно, что число неравных правильных треугольников равно 0 или 1, а число неравных равнобедренных равно $(n - 1)/2$ или $(n/2) - 1$, т. е. $N_1 = 1 - c$, $N_1 + N_2 = (n-2+d)/2$, где c и d равны 0 или 1. Поэтому $12N = 12(N_1 + N_2 + N_3) = 2(N_1 + 3N_2 + 6N_3) + 6(N_1 + N_2) + 4N_1 = (n - 1)(n - 2) + 3(n - 2 + d) + 4(1 - c) = n^2 + 3d - 4c$. Так как $|3d - 4c| < 6$, то N совпадает с ближайшим к $n^2/12$ целым числом.

Другое решение : переборное за O(N)

Вообще в любой задаче первая мысль должна быть: а нельзя ли перебрать. Ну и что перебрать) И еще мысль: а нельзя ли посчитать от предыдущих. И еще одна: а нельзя ли получить ответы перебором и подобрать зависимость или формулу)

Тройка чисел a<=b<=c задает уникальным образом треугольник. Зафиксируем первую вершину. Будем строить треугольник по часовой стрелке, по мере увеличения длин сторон. Зная длину первой стороны, не сложно найти все возможные значения для второй (а третья уже однозначна задается).

```
#include <stdio.h>
#include <sstream>
#include <iostream>
#include <string>
#include <algorithm>
#include <vector>
#include <list>
#include <iomanip>
#include <map>
#include <set>
#include <cmath>
#include <queue>
#include <cassert>
#include <string.h>
using namespace std;
#pragma comment(linker, "/STACK:20000000")

typedef vector<int> vi;
#define sz(a) int((a).size())
#define all(c) (c).begin(),(c).end()
```

```
int main()
{

	int n;
	cin >> n;
	long long res = 0;

	for (int a=1; 3*a <= n; ++a){
		int left = n - 3*a;
		res += left/2 + 1;
	}

	cout << res << endl;

	return 0;
}
```

E Наборы чисел

Здесь очевидно, что количество чисел не равных 1 максимум 8 (2^8 = 256). Поэтому пишем рекурсивный перебор

Решение C++

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <cstdio>
#include <cmath>
#include <string>
#include <algorithm>
#include <vector>
#include <queue>
#include <stack>
#include <vector>
#include <set>
#include <map>
#include <sstream>
#include <cassert>
using namespace std;
#pragma comment(linker, "/STACK:20000000")

typedef vector<int> vi;
#define sz(a) int((a).size())
#define all(c) (c).begin(),(c).end()


long long t[100005];
int mn[100005];
int n;
int res;
int mx;
// сумма, произведение, последнее число, сколько осталось поставить
void go(long long s, long long p, int l, int n) {
    if (n < 0) return;
    if (n == 0) {
        if (s == p) {
```

```
            res++;
            mx = max(mx, l);
        }
        return;
    }

    for (int i = l; i < 550; i++) {
        if (p * i - s - i <= n)
            go(s + i, p * i, i, n - 1);
        else break;
    }
}


int main() {


    cin >> n;

    res = 0;
    go(0, 1, 1, n);
    cout << res << endl;
}
```

F Самое маленькое число

В этой задаче можно было просто посчитать все возможные варианты ответов и выбрать минимальный. Например запускаясь от набора и, пробегаясь по всем парам чисел в нем, применять очередную операцию и рекурсивно запускаться от нового набора. Когда число остается одно, сравнить его с минимальным уже полученным, и, если нужно, изменить минимальное.

Решение C++

```
#include<bits/stdc++.h>
typedef long long ll;
using namespace std;
const ll INF=1e15;
ll x;
vector<ll> v;
char w[4];
ll ans;
void dfs(ll step)
{
      if(step==3)
      {
            ans=min(ans,v[0]);
            return;
      }
      vector<ll> s;
      s=v;
      for(ll i=0;i<v.size();i++)
      {
```

```
            ll now=v[i];
            for(ll j=i+1;j<v.size();j++)
            {
                  ll sum,renow=v[j];
                  if(w[step]=='*') sum=now*renow;
                  else sum=now+renow;
                  v.erase(v.begin()+j);
                  v.erase(v.begin()+i);
                  v.push_back(sum);
                  dfs(step+1);
                  v=s;
            }
      }
      return;
}
int main()
{
      ans=INF;
      for(ll i=0;i<=3;i++) cin>>x,v.push_back(x);
      cin>>w[0]>>w[1]>>w[2];
      dfs(0);
      printf("%lld\n",ans);
      return 0;
}
```

Решение Python

```
from itertools import permutations

nums = list(map(int, input().split()))
o = input().split()

def f(op, a, b):
  return a * b if op == "*" else a + b

res = 1000 ** 4
for a, b, c, d in permutations(nums):
  res = min(res, min(f(o[2], f(o[1], f(o[0], a, b), c), d), \
                     f(o[2], f(o[0], a, b), f(o[1], c, d))))
print(res)
```

G Очень занимательная игра

Перебираем какое число мы поставим(I) первым оно будет от 1 до min(mod-1,A) потому-что дальше mod перебирать нет смысла, числа по модулю будут повторяться. Дальше домножим наше первое число на (10^9)%mod и возьмем произведение по модулю, пускай оно будет X тогда число I подходит если X!=0 и X+B<mod, что вроде-бы очевидно.

Решение C++

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    long long a,b,mod;
    cin>>a>>b>>mod;

    for(long long i=1; i<mod && i<=a; i++)
    {
        if((mod-(i*1000000000)%mod)%mod  >b)
        {
            printf("1 %09lld\n",i);
            return 0;
        }
    }
    cout<<"2\n";
}
```

H Чудо-комната

Без ограничения общности можем считать, что $a \le b$.

Во первых, надо рассмотреть случай, в котором уже можно поселить всех людей. Если $6 \cdot n \le a \cdot b$, то ответ $a \cdot b$ a b.

В ином случае нам нужно увеличить какую-то сторону(возможно обе). Делать это будем так: переберем меньшую сторону комнаты new_a ($a \le new_a \le \lceil \sqrt{6 \cdot n} \rceil$), после того, как мы зафиксировали new_a, другую сторону можно посчитать как $new_b = \lceil \frac{6 \cdot n}{new_a} \rceil$. По всем таким new_a и new_b, если $b \le new_b$, выбираем такие, произведение которых наименьшее.

Понятно, что рассматривать $new_a > \lceil \sqrt{6 \cdot n} \rceil$ не имеет смысла, так как мы можем просто ее уменьшить и получить комнату меньшей площади. При этом она будет удовлетворять условию, так как $6 \cdot n \le (\lceil \sqrt{6 \cdot n} \rceil)^2 \le new_a \cdot new_b$.

Также нужно внимательно следить за тем, чтобы значения не вылезли за используемый тип данных.

Бонус: можете ли вы уменьшить верхнюю оценку перебора меньшей стороны?

Асимптотика: $O(\sqrt{n})$

Решение C++

```
#include <iostream>

using namespace std;

int main()
{
    ios::sync_with_stdio(0);
    long long n, a, b;

    cin >> n >> a >> b;
    if (6*n <= a*b)
        cout << a*b << "\n" << a << " " << b << "\n";
    else {
        bool f = 0;
        if (a > b)
```

```
        {
            swap(a, b);
            f = 1;
        }

        long long SQ = 1e18, a1, b1, tmpb;
        for(long long i = a; i*i <= 6*n; ++i) {
            tmpb = 6*n / i;
            if (tmpb * i < 6*n) tmpb++;

            if (tmpb < b) continue;

            if (tmpb * i < SQ) {
                SQ = tmpb * i;
                a1 = i;
                b1 = tmpb;
            }
        }

        if (f)
            swap(a1, b1);

        cout << SQ << "\n" << a1 << " " << b1 << "\n";
    }

    return 0;
}
```