Олимпиада школьников
по прикладной математике и информатике
факультета вычислительной математики и кибернетики
Московского государственного университета
имени М. В. Ломоносова

Очный тур (16 апреля 2016 года)

8-9 классы

Задачи по математике

1. Ребус. Расшифруйте ребус (одинаковые буквы соответствуют одинаковым цифрам, разные — разным):

$$\text{В} \cdot \text{М} \cdot \text{К} \cdot \text{МГУ} = 2016.$$

Ответ: $3 \cdot 1 \cdot 4 \cdot 168 = 2016$ или $4 \cdot 1 \cdot 3 \cdot 168 = 2016$.

Решение. Поскольку $\text{МГУ} \geqslant 102$, то $\text{В} \cdot \text{М} \cdot \text{К} \leqslant \frac{2016}{102} < 20$. Заметим, что $2016 = 2^5 \cdot 3^2 \cdot 7$. Осталось перебрать натуральные делители числа 2016, которые можно представить в виде произведения трех различных однозначных натуральных чисел (делителей того же числа 2016), причем сумма этих трех чисел не должна превосходить числа 19. Произведения (в каком-то порядке) $7 \cdot 2 \cdot 1$, $9 \cdot 2 \cdot 1$, $3 \cdot 2 \cdot 1$ и $4 \cdot 2 \cdot 1$ не подходят в качестве $\text{В} \cdot \text{М} \cdot \text{К}$, т. к. в числе МГУ все цифры разные, а $\text{В} \cdot \text{М} \cdot \text{К} = 8 \cdot 2 \cdot 1$ влечет $\text{МГУ} = 126$ и не подходит по причине того, что буква Г не встречается в ВМК. Остается только один вариант — произведение $3 \cdot 4 \cdot 1$, который и дает указанный выше ответ.

2. Азартные игры. Студент играет в игру с профессором. На доске записано уравнение $x^2 + 2x + 10 = 0$. Сначала студент увеличивает или уменьшает коэффициент при x на 2. Затем профессор увеличивает или уменьшает свободный член на 2016. Потом студент снова увеличивает или уменьшает коэффициент при x на 2, и т. д. Студент выигрывает, если в какой-нибудь момент в процессе игры уравнение будет иметь целый корень. Может ли профессор помешать ему выиграть?

Ответ: да, сможет.

Решение. На самом деле, профессор может ходить как угодно — студент не выиграет никогда. Действительно, предположим, что в какой-то момент игры на доске возникло уравнение, имеющее целый корень. Ясно, что уравнение квадратное, и сумма корней равна целому числу (коэффициенту при x, взятому с противоположным знаком), поэтому второй корень тоже целый. Поскольку $10 + 2016n$ (равное по теореме Виета произведению корней) при любом целом n имеет остаток 2 при делении на 4, то один корень четный, а

другой нечетный. Значит, сумма корней нечетна. А это противоречит тому, что $2+2k$ является четным при любом целом k.

3. Радикальное построение. Дан отрезок длины $\sqrt{3}+\sqrt{5}$. Можно ли с помощью циркуля и линейки построить отрезок длины 1?

Ответ: да, можно.

Решение. Пусть $\sqrt{3}+\sqrt{5}=p$. Используя стандартное построение прямоугольного треугольника по двум катетам, можно легко построить отрезки длины $p\sqrt{2}$, а затем $p\sqrt{3}$. Построив прямоугольный треугольник по двум катетам p и $2p$, получим гипотенузу длины $p\sqrt{5}$. Откладывая отрезки длины $p\sqrt{3}$ и $p\sqrt{5}$ рядом на одной прямой, имеем отрезок длины $p(\sqrt{3}+\sqrt{5})=p^2$. Наконец, откладывая отрезки p^2 и p на одной стороне произвольного угла (от вершины), и отрезок p — на другой стороне этого угла и проводя параллельные прямые, получим отрезок длины $\frac{p\cdot p}{p^2}=1$.

4. Мерцающий кубик. В вершинах куба находятся числа. Каждую секунду каждое число заменяется на среднее геометрическое чисел, стоящих в соседних вершинах (среднее геометрическое трех чисел a, b, c равно величине $\sqrt[3]{abc}$). Ровно через минуту выяснилось, что расстановка чисел совпала с исходной. Обязательно ли все исходные числа были равны?

Ответ: нет, не обязательно.

Решение. Раскрасим вершины куба в два цвета так, чтобы концы каждого ребра были окрашены в разные цвета. Поместим в каждую вершину первого цвета число 1, а в каждую вершину второго цвета — число -1. Очевидно, через каждые 2 секунды расстановка чисел повторяется. Значит, она повторится и через 60 секунд.

5. Загадочная последовательность. В последовательности $2, 3, 0, 1, \ldots$ каждый член, начиная с пятого, равен последней цифре суммы четырех предыдущих. Встретится ли в этой последовательности четверка подряд идущих чисел $2, 0, 1, 6$?

Ответ: да, встретится.

Решение. Заметим, что каждый член, начиная с пятого, полностью определяется четырьмя предыдущими. Более того, каждый член полностью определяется также и четырьмя *последующими* членами. Поскольку количество различных четверок последовательных членов конечно (их не больше, чем 10^4), последовательность будет периодической. Остается заметить, что если идти не вперед, а назад, то очень скоро мы дойдем до искомой четверки: $\ldots 2, 0, 1, 6, 9, 6, 2, 3, 0, 1$. В силу периодичности, фрагмент $2, 0, 1, 6$ встретится и впереди (причем бесконечное число раз).

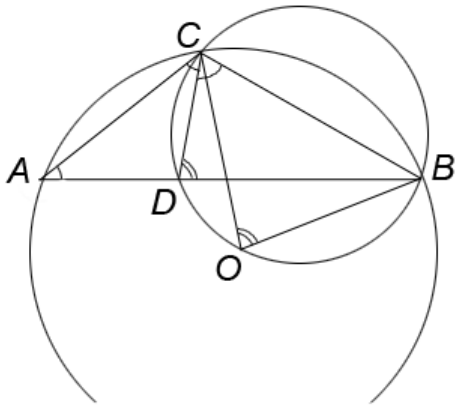
6. Разыскивается треугольник. Найти треугольник, длины всех сторон и длины всех высот которого – целые числа. В ответе указать длины сторон.

Пример правильного ответа: 15, 20, 25.

Легко видеть, что треугольник со сторонами $15, 20, 25$ является прямоугольным (он подобен египетскому треугольнику со сторонами $3, 4, 5$). Две высоты совпадают с катетами, а высота, проведенная к гипотенузе, равна $(15 \cdot 20)/25 = 12$. Существуют и другие примеры. Например, треугольник со сторонами $65, 156, 169$.

7. Равные произведения. В треугольнике ABC угол C в два раза больше угла A. Окружность, проходящая через точки B, C и центр описанной около треугольника ABC окружности, пересекает сторону AB в точке D. Доказать, что $AC \cdot BD = AD \cdot BC$.

Решение. Обозначим $\angle BAC = \alpha$ и $\angle ACB = 2\alpha$. Пусть O — центр окружности, описанной около треугольника ABC. Тогда $\angle BAC$ — вписанный угол, а $\angle BOC$ — центральный угол, причем оба они опираются на одну дугу BC (здесь важно, что в треугольнике ABC угол A не может быть тупым, поскольку он меньше угла C). Поэтому $\angle BOC = 2\angle BAC = 2\alpha$.



Далее, $\angle BDC = \angle BOC = 2\alpha$ как равные вписанные углы в окружности, проходящей через точки B, C и O. Кроме того, $\angle BDC$ — внешний угол треугольника ACD, поэтому $\angle ACD = \angle BDC - \angle BAC = \alpha$. Значит, CD — биссектриса треугольника ABC, поэтому по основному свойству биссектрисы имеем $\frac{AD}{BD} = \frac{AC}{BC}$, а значит, $AC \cdot BD = AD \cdot BC$, что и требовалось доказать.

Задачи на программирование

В задачах на программирование требуется обосновать корректность предлагаемого участником олимпиады алгоритма решения задачи и предложить реализацию этого алгоритма на одном из языков программирования (допускаются также блок-схемы и иные точные описания алгоритмов). В ряде случаев при оценке таких задач учитывается эффективность алгоритма. Решения приведены на языке Python.

8. Палиндром с ошибкой. Наборщик пропустил одну букву при наборе некоторого слова. Проверить, могло ли это слово в источнике быть палиндромом.

Ввести слово, состоящее из маленьких латинских букв. Если добавлением одной буквы это слово можно превратить в палиндром, вывести этот палиндром. В противном случае вывести «Нет».

Примеры:

«agerteanonaetrga» — agerteanonaetrega «paaxcexuluxecdaap» — Нет

Решение. Пояснение. Если отбрасывать с двух сторон по одинаковой букве, возможны следующие варианты.

1. Получится пустое слово. Это значит, что палиндром возможен с любой буквой посредине.

2. Получится слово из одной буквы. Это значит, что палиндром возможен с двумя такими буквами посредине.

3. Получится слово, первая буква которого не равна последней.

3.1 Пропущена последняя буква этого слова в начале этого слова. При добавлении должен получиться строгий палиндром.

3.2 Пропущена первая буква в конце. При добавлении должен получиться строгий палиндром.

```
#!/usr/bin/env python3
Root = Word = input()
l = len(Word)//2

# Парные буквы выбрасываем
while len(Root)>1 and Root[0] == Root[-1]:
    Root = Root[1:-1]
lr = len(Root)
if lr == 0:
    # Пропущена средняя буква (любая)
    print(Word[:l]+Word[0]+Word[l:])
elif lr == 1:
    # Пропущена одна из двух средних букв
    print(Word[:l]+Root+Word[l:])
else:
    # Пропущена последняя буква фрагмента
    if Root[1:] == Root[-1:0:-1]:
        print(Word[:l+lr-1]+Root[0]+Word[l+lr-1:])
    # Пропущена первая буква фрагмента
    elif Root[:-1] == Root[-2::-1]:
        print(Word[:l-lr//2]+Root[-1]+Word[l-lr//2:])
    else:
        print("NO")
```

9. Ожерелье шамана. Ожерелье шамана Кынлабаса представляет собой замкнутое верёвочное кольцо, на которое нанизаны бусины. Бусины соответствуют четырём стихиям — Воде, Земле, Воздуху и Огню. Бусы одного типа, а также бусы Воды и Огня, и бусы Воздуха и Земли враждебны друг другу, остальные сочетания нейтральны. Колдовскую силу ожерелье обретает, если

враждебные бусы в нём не соседствуют. Зная требуемое количество бусин каждого типа, ответить, существует ли колдовское (в кынлабасовом смысле) ожерелье.

Ввести 4 натуральных числа, обозначающих количества бусин Воды, Земли, Воздуха и Огня соответственно. Вывести «Да», если можно создать ожерелье с такими количествами бусин этих типов, в котором враждебные бусины не соседствуют друг с другом, и «Нет» в противном случае.

Примеры:

2 2 2 2 — «Да»

9 1 1 1 — «Нет»

Решение.

```
#!/usr/bin/env python3
'''
Пояснение.

Чего делать нельзя - перебирать все сочетания и проверять, годятся ли они.

Что можно было делать - рекурсивное построение с отсечением (см. ниже).

Что на самом деле - бусины враждебных элементалей функционально неразличимы при
данных условиях, так что ответ "Да" равнозначен утверждению "бусин Воды и Огня
вместе столько же, сколько бусин Земли и Воздуха вместе".
'''

W,E,A,F = eval(input())
NUM=4

# Так как частный случай задачи прост, решим более общий,
# в котром совиместимость бусин определяется функцией
def compat(i,j):
    '''В нашем случае совместимость - неодинаковая чётность в списке'''
    return i%2 != j%2

# Поскольку бусы кольцевые, начинать можно с любой бусины - например, с Воды
# В стеке хранятся тройки:
#   тип последней добавленной бусины
#   остаток бусин (список из 4 чисел)
#   тип бусины, которую мы попытаемся добавить к данной в следующий раз
Stack=[[0,[W-1,A,F,E],0]]
while Stack:
    C, S, N = Stack[-1]
    if S[0]==S[1]==S[2]==S[3]==0:        # Бусины кончились?
        if compat(C,0):                  # Последняя бусина совместима с первой?
            print("Да")
            break
    for i in range(N,NUM):               # Попытаемся добавить бусину
        if compat(i,C) and S[i]>0:       # Подходит
            Z=S.copy()
```

```
            Z[i]-=1                     # Остаток бусин
            Stack[-1][-1]=i+1           # Потом продолжим искать со следующей
            Stack.append([i,Z,0])       # Новое состояние
            break                       # Было добавление, обходим else:
    else:                               # else к оператору for: добавить не вышло
        Stack.pop()                     # Текущая бусина оказалась неудачной
else:                                   # else к оператору for: стек закончился
    print("Нет")
```

Олимпиада школьников
по прикладной математике и информатике
факультета вычислительной математики и кибернетики
Московского государственного университета
имени М. В. Ломоносова

Очный тур (16 апреля 2016 года)

10 класс

Задачи по математике

1. Азартные игры. Школьник играет в игру с академиком. На доске записан многочлен $x^2+9x+2010$. Сначала академик прибавляет или вычитает число 2016. Затем школьник дописывает $+x$ или $-x$. Потом академик снова прибавляет или вычитает число 2016, и т. д. Академик выигрывает, если в какой-нибудь момент в процессе игры многочлен будет иметь целый корень. Сможет ли школьник помешать ему выиграть?

Ответ: да, сможет.

Решение. Пусть школьник первым ходом пишет $-x$, вторым ходом дописывает $+x$, третьим ходом опять пишет $-x$, и т. д. При этом после приведения подобных слагаемых на доске всякий раз будет получаться квадратный трехчлен, у которого коэффициент при x равен 8 или 9.

Покажем, что академик не сможет выиграть. Предположим, что в какой-то момент игры на доске возникло уравнение, имеющее целый корень. По теореме Виета сумма корней трехчлена равна коэффициенту при x, взятому с противоположным знаком, поэтому второй корень тоже целый. Свободный член $2010+2016n$, равный произведению корней, при любом целом n делится на 2, но не делится на 4. Значит, корни разной четности, и их сумма нечетна. С другой стороны, $2010 + 2016n$ при любом целом n делится на 3, но не делится на 9. Значит, один корень делится на 3, а другой — нет. Поэтому сумма корней не делится на 3.

В итоге, если трехчлен имеет целые корни, то их сумма не делится ни на 2, ни на 3. Это противоречит тому, что коэффициент при x всегда на протяжении игры равен 8 или 9.

2. Радикальное построение. Дан отрезок длины $\sqrt{3}+\sqrt{5}+\sqrt{7}$. Можно ли с помощью циркуля и линейки построить отрезок длины 1?

Ответ: да, можно.

Решение. Пусть $\sqrt{3} + \sqrt{5} + \sqrt{7} = p$. Используя стандартное построение прямоугольного треугольника по двум катетам, можно легко построить отрезки длины $p\sqrt{2}$, а затем $p\sqrt{3}$. Построив прямоугольный треугольник по двум катетам p и $2p$, получим гипотенузу длины $p\sqrt{5}$. Аналогично, построив прямоугольный треугольник по двум катетам $p\sqrt{3}$ и $2p$, получим гипотенузу

длины $p\sqrt{7}$. Откладывая отрезки длины $p\sqrt{3}$, $p\sqrt{5}$ и $p\sqrt{7}$ рядом на одной прямой, имеем отрезок длины $p(\sqrt{3}+\sqrt{5}+\sqrt{7}) = p^2$. Наконец, откладывая отрезки p^2 и p на одной стороне произвольного угла (от вершины), и отрезок p — на другой стороне этого угла и проводя параллельные прямые, получим отрезок длины $\frac{p \cdot p}{p^2} = 1$.

3. Мерцающий кубик. В вершинах куба находятся числа. Каждую секунду каждое число заменяется на среднее геометрическое чисел, стоящих в соседних вершинах (среднее геометрическое трех чисел a, b, c равно величине $\sqrt[3]{abc}$). Ровно через минуту выяснилось, что расстановка чисел совпала с исходной. Обязательно ли все исходные числа были равны?

Ответ: нет, не обязательно.

Решение. Раскрасим вершины куба в два цвета так, чтобы концы каждого ребра были окрашены в разные цвета. Поместим в каждую вершину первого цвета число 1, а в каждую вершину второго цвета — число -1. Очевидно, через каждые 2 секунды расстановка чисел повторяется. Значит, она повторится и через 60 секунд.

4. Загадочная последовательность. В последовательности $2, 3, 0, 1, \ldots$ каждый член, начиная с пятого, равен последней цифре суммы четырех предыдущих. Встретится ли в этой последовательности четверка подряд идущих чисел $2, 0, 1, 6$?

Ответ: да, встретится.

Решение. Заметим, что каждый член, начиная с пятого, полностью определяется четырьмя предыдущими. Более того, каждый член полностью определяется также и четырьмя *последующими* членами. Поскольку количество различных четверок последовательных членов конечно (их не больше, чем 10^4), последовательность будет периодической. Остается заметить, что если идти не вперед, а назад, то очень скоро мы дойдем до искомой четверки: $\ldots 2, 0, 1, 6, 9, 6, 2, 3, 0, 1$. В силу периодичности, фрагмент $2, 0, 1, 6$ встретится и впереди (причем бесконечное число раз).

5. Разыскивается параллелограмм. Найти параллелограмм $ABCD$, не являющийся ни ромбом, ни прямоугольником, у которого длины всех сторон и длины всех диагоналей – целые числа. В ответе указать длины отрезков AB, BC, AC, BD.

Пример правильного ответа: 11, 12, 13, 19.

Пример правильного ответа: 11, 23, 20, 30.

Проверку выполнения условия задачи удобно производить так. Рассмотрим треугольник со сторонами $a = 11$, $b = 12$, $c = 13$ и найдем его медиану, проведенную к стороне c, по формуле $m_c = \frac{1}{2}\sqrt{2(a^2+b^2)-c^2} = \frac{19}{2}$. Удваивая эту медиану, получим параллелограмм со смежными сторонами 11 и 12 и диагоналями 13 и 19. Он не является прямоугольником ввиду того, что его

диагонали различны.

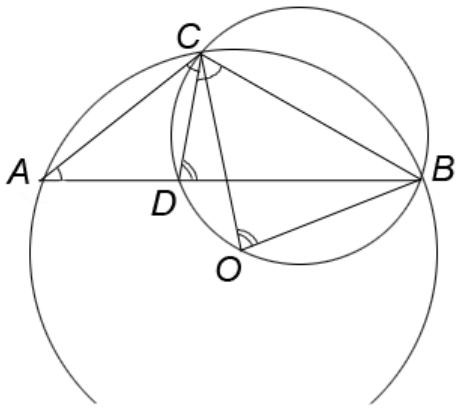
Аналогично можно выполнить проверку и для второго примера.

6. Прогрессивный трехчлен. Квадратный трехчлен $ax^2 + bx + c$ имеет два корня x_1 и x_2. Незнайка заметил, что все попарно различные элементы перечня a, b, c, x_1, x_2 в некотором порядке образуют арифметическую прогрессию из целых чисел. Возможно ли это?

Ответ: возможно. Например, трехчлен $x^2 + 0x - 1$ имеет корни $x_{1,2} = \pm 1$, и $\{a, b, c, x_1, x_2\} = \{-1; 0; 1\}$.

7. Равные произведения. В треугольнике ABC угол C в два раза больше угла A. Окружность, проходящая через точки B, C и центр описанной около треугольника ABC окружности, пересекает сторону AB в точке D. Доказать, что $AC \cdot BD = AD \cdot BC$.

Решение. Обозначим $\angle BAC = \alpha$ и $\angle ACB = 2\alpha$. Пусть O — центр окружности, описанной около треугольника ABC. Тогда $\angle BAC$ — вписанный угол, а $\angle BOC$ — центральный угол, причем оба они опираются на одну дугу BC (здесь важно, что в треугольнике ABC угол A не может быть тупым, поскольку он меньше угла C). Поэтому $\angle BOC = 2\angle BAC = 2\alpha$.



Далее, $\angle BDC = \angle BOC = 2\alpha$ как равные вписанные углы в окружности, проходящей через точки B, C и O. Кроме того, $\angle BDC$ — внешний угол треугольника ACD, поэтому $\angle ACD = \angle BDC - \angle BAC = \alpha$. Значит, CD — биссектриса треугольника ABC, поэтому по основному свойству биссектрисы имеем $\frac{AD}{BD} = \frac{AC}{BC}$, а значит, $AC \cdot BD = AD \cdot BC$, что и требовалось доказать.

Задачи на программирование

В задачах на программирование требуется обосновать корректность предлагаемого участником олимпиады алгоритма решения задачи и предложить реализацию этого алгоритма на одном из языков программирования (допускаются также блок-схемы и иные точные описания алгоритмов). В ряде случаев при оценке таких задач учитывается эффективность алгоритма. Решения приведены на языке Python.

8. Неполное раскрытие. Участники космических гонок на выживание хотят выяснить, кто из них ближе к Центру Галактики. Но чтобы не раскрывать своего положения полностью, они сообщают только целую часть своих координат (дробную, даже очень большую, скрывают). Можно ли по имеющимся данным выявить текущего лидера космических гонок на выживание? (Предполагается, что в Галактике задана единая для всех участников прямоугольная трехмерная система координат).

Ввести 100000 троек целых чисел — округлённые координаты участников.

Вывести номер участника, который находится ближе всех к точке $(0, 0, 0)$ — Центру Галактики, или 0, если выяснить это нельзя. Первый участник имеет номер 1.

Примеры (для трёх участников):

- $11, -11, 11;\ -20, 10, 20;\ 13, 14, -15; -1$
- $8, 15, -7;\ -9, 16, 15;\ -10, -11, 12; -0$

Решение.

```
#!/usr/bin/env python3

"""
Пояснение. Сообщаемые двумя участниками координаты могут давать одно
соотношение расстояний до центра, в то время как в реальности оно может быть
другим. Например, участник, находящийся внутри куба со стороной 2 и центром
в Центре галактики, сообщит координаты (0,0,0), даже если близок к углу куба
(расстояние - квадратный корень из трёх), а у участника, находящегося вне этого
куба, как минимум одна из координат будет не меньше 1, даже если они близок
(снаружи) к середине грани (расстояние - 1).

Для каждого участника существует минимально возможное удаление от центра
(очевидно, равное расстоянию от центра до сообщаемой точки x,y,z) и максимально
возможное (не превосходящее расстояния до |x|+1,|y|+1,|z|+1).
Определить лидера гонок - участника с наименьшим минимально возможным удалением
- можно, только если его максимально возможное удаление не больше минимально
возможного для остальных участников. """

# Введём строку, отрежем от неё последний символ (';')
s = input()[:-1]
# разобьём её на части, разделённые ';'
s=s.split(';')
# Формат через запятую - стандартный для Python3, превратим в тройки координат
s = [eval(c) for c in s]
# Возьмём абсолютные величины координат
Racers = [(abs(x), abs(y), abs(z)) for x,y,z in s]

# Список квадратов минимальных и максимальных  расстояний от центра
# (корень вычислять незачем)
MinMax = [((x**2+y**2+z**2),((x+1)**2+(y+1)**2+(z+1)**2)) for x,y,z in Racers]
```

```
# Данные предполагаемого лидера
Min, Max = min(MinMax)

# Номер этого участника (можно было бы найти вместе с данными)
N = MinMax.index((Min, Max))

# Все участники, которые могут быть впереди "лидера"
Maybe = [(m,M) for m,M in MinMax if m<Max]

# В этом списке должен быть один только лидер
if len(Maybe)>1:
    print(0)
else:
    print(N+1)
```

9. Ожерелье шамана. Ожерелье шамана Кынлабаса представляет собой замкнутое верёвочное кольцо, на которое нанизаны бусины. Бусины соответствуют четырём стихиям — Воде, Земле, Воздуху и Огню. Бусы одного типа, а также бусы Воды и Огня, и бусы Воздуха и Земли враждебны друг другу, остальные сочетания нейтральны. Колдовскую силу ожерелье обретает, если враждебные бусы в нём не соседствуют. Зная требуемое количество бусин каждого типа, ответить, существует ли колдовское (в кынлабасовом смысле) ожерелье.

Ввести 4 натуральных числа, обозначающих количества бусин Воды, Земли, Воздуха и Огня соответственно. Вывести «Да», если можно создать ожерелье с такими количествами бусин этих типов, в котором враждебные бусины не соседствуют друг с другом, и «Нет» в противном случае.

Примеры:

2 2 2 2 — «Да»

9 1 1 1 — «Нет»

Решение.

```
#!/usr/bin/env python3
'''
Пояснение.

Чего делать нельзя - перебирать все сочетания и проверять, годятся ли они.

Что можно было делать - рекурсивное построение с отсечением (см. ниже).

Что на самом деле - бусины враждебных элементалей функционально неразличимы при
данных условиях, так что ответ "Да" равнозначен утверждению "бусин Воды и Огня
вместе столько же, сколько бусин Земли и Воздуха вместе".
'''

W,E,A,F = eval(input())
NUM=4
```

```
# Так как частный случай задачи прост, решим более общий,
# в котром совиместимость бусин определяется функцией
def compat(i,j):
    '''В нашем случае совместимость - неодинаковая чётность в списке'''
    return i%2 != j%2

# Поскольку бусы кольцевые, начинать можно с любой бусины - например, с Воды
# В стеке хранятся тройки:
#   тип последней добавленной бусины
#   остаток бусин (список из 4 чисел)
#   тип бусины, которую мы попытаемся добавить к данной в следующий раз
Stack=[[0,[W-1,A,F,E],0]]
while Stack:
    C, S, N = Stack[-1]
    if S[0]==S[1]==S[2]==S[3]==0:        # Бусины кончились?
        if compat(C,0):                  # Последняя бусина совместима с первой?
            print("Да")
            break
    for i in range(N,NUM):               # Попытаемся добавить бусину
        if compat(i,C) and S[i]>0:       # Подходит
            Z=S.copy()
            Z[i]-=1                      # Остаток бусин
            Stack[-1][-1]=i+1            # Потом продолжим искать со следующей
            Stack.append([i,Z,0])        # Новое состояние
            break                        # Было добавление, обходим else:
    else:                                # else к оператору for: добавить не вышло
        Stack.pop()                      # Текущая бусина оказалась неудачной
else:                                    # else к оператору for: стек закончился
    print("Нет")
```