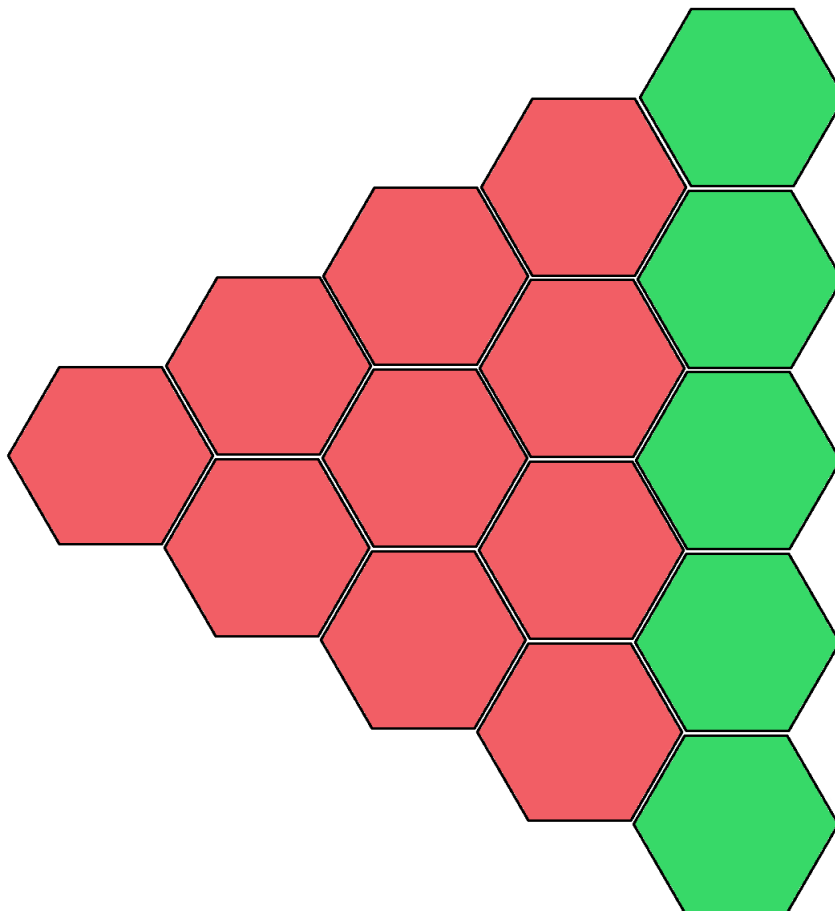


Задача 1. Треугольник

Для того, чтобы набрать 50 баллов за задачу можно заранее посчитать значения для всех значений от 1 до 7 и вывести ответ с использованием условного оператора

```
n = int(input())
if n == 1:
    print(1)
elif n == 2:
    print(3)
elif n == 3:
    print(6)
elif n == 4:
    print(10)
elif n == 5:
    print(15)
elif n == 6:
    print(21)
elif n == 7:
    print(28)
```

Заметим, что ответ для каждого следующего значения n отличается от предыдущего ответа на величину n . Действительно, если известно количество шестиугольников для i , то в треугольник с размером стороны $i + 1$ войдут все эти шестиугольники и еще один ряд. Например, на следующем рисунке изображен переход от длины стороны 4 к длине стороны 5.



Таким образом, чтобы получить ответ для любого значения n нужно посчитать сумму чисел от 1 до n . Сделать это можно с помощью цикла:

```
n = int(input())
s = 0
for i in range(1, n+1):
    s += i
print(s)
```

Также можно посчитать ответ с использованием формулы суммы арифметической прогрессии:

$$Ans = \frac{n \cdot (n + 1)}{2}.$$

Программа на языке Python:

```
n = int(input())
print((1 + n) * n // 2)
```

Задача 2. Шоколадка

Один из возможных вариантов решения данной задачи - перебрать все возможные варианты выполнения разлома по горизонтали и вертикали. Для этого переберем значение в месте разлома i выполняемого по вертикали и для каждого такого значения будем определять сколько долек достанется Саше $n * i$ и сколько останется у Маши $n * m - n * i = n * (m - i)$.

После этого таким же образом проверим горизонтальные разломы шоколадки. Выберем наибольшее из чисел и выведем на экран.

```
n = int(input())
m = int(input())

c1 = 0
for i in range(1, m):
    if c1 < n * i < n * (m - i):
        c1 = n * i

for i in range(1, n):
    if c1 < m * i < m * (n - i):
        c1 = m * i

print(c1, n * m - c1)
```

Решение можно оптимизировать если перебирать значения i не до m и n , а до $m/2$ и $n/2$ соответственно, так как, очевидно, если $i > m/2$ или $i > n/2$, то Саша получит больше половины долек, а это не подходит.

Заметим, что можно начать перебор не от 1, а от половин с каждой стороны.

```
n = int(input())
m = int(input())

c1 = 0
for i in range(m // 2, 0, -1):
    if c1 < n * i < n * (m - i):
        c1 = n * i
        break

for i in range(n // 2, 0, -1):
    if c1 < m * i < m * (n - i):
        c1 = m * i
```

break

```
print(c1, n * m - c1)
```

Эффективное решение данной задачи не использует циклов. Заметим, что размер стороны, по которой выполняется разлом - чётное число, то разлом нужно выполнять так, чтобы Саше досталось на 1 меньше половины по этой стороне. Если же размер стороны - нечётное число, то просто поделим её длину нацело на 2.

```
n = int(input())
m = int(input())

c1 = 0
if m % 2 == 0:
    c1 = n * (m // 2 - 1)
else:
    c1 = n * (m // 2)

if n % 2 == 0:
    c1 = max(c1, m * (n // 2 - 1))
else:
    c1 = max(c1, m * (n // 2))

print(c1, n * m - c1)
```

Это же можно записать короче, если использовать остаток от деления в формуле:

```
n = int(input())
m = int(input())

v1 = n * (m // 2 - 1 + m % 2)
v2 = m * (n // 2 - 1 + n % 2)

c1 = v1
if c1 < v2:
    c1 = v2

print(c1, n * m - c1)
```

Получившиеся значения могут быть больше, чем возможное значение 32-битной целочисленной переменной, поэтому для полного решения в других языках необходимо использовать 64-битные числа (тип int64 в языке Pascal, тип long long в C и C++, тип long в Java и C#).

Задача 3. Участки пайщикам!

Заметим, что в первой группе тестов $m = 2$, а значит для решения задачи нужно реализовать алгоритм поиска второго максимума.

Для получения частичного балла за данную задачу можно реализовать следующий подход: считаем данные в список, а затем будем выбирать m раз максимальное число и удалять его из списка.

```
from sys import stdin
inp = stdin.readline

def solve():
    n, m = map(int, input().split())
    a = [ ]
    for i in range(n):
```

```
h, w = map(int, input().split())
a.append((h * w, i + 1))

for i in range(m):
    t = 0
    for j in range(1, len(a)):
        if a[j][0] > a[t][0]:
            t = j
    if i == m-1:
        print(a[t][1])
    else:
        del a[t]

solve()
```

Для получения полного балла необходимо использовать сортировку данных в списке по убыванию значений площади, но при этом по возрастанию номеров участков

```
import sys

def solve():
    n, m = map(int, sys.stdin.readline().split())
    a = []
    for i in range(n):
        h, w = map(int, sys.stdin.readline().split())
        a.append((h * w, -(i + 1)))
    a.sort(reverse=True)
    print(abs(a[m-1][1]))

solve()
```

Задача 4. Биржевой брокер

Для того, чтобы получить частичный балл на данной задаче можно реализовать следующую идею: добавим все данные в массив, а затем для каждого элемента будем просматривать $m - 1$ элементов назад и находить наименьшее и наибольшее отклонение. После чего проверяем условия для заданных граничных значений. Ниже пример программы, реализующий данный алгоритм на языке C++

```
#include <iostream>
#include <vector>

void solve() {
    int n, m, u, d;
    std::cin >> n >> m >> u >> d;
    std::vector<int> a;
    for(size_t i=0; i<n; ++i) {
        int x;
        std::cin >> x;
        a.emplace_back(x);
    }

    for(int i=1; i<n; ++i) {
        int mn = (int)1e9;
        int mx = -mn;
```

```
        for(int j=std::max(0, i+1-m); j<i; ++j) {
            mn = std::min(mn, a[i]-a[j]);
            mx = std::max(mx, a[i]-a[j]);
        }
        mn = -mn;
        if (mx >= u && mn >= d) {
            std::cout << i + 1 << "?\n";
        } else if (mx >= u) {
            std::cout << i + 1 << "+\n";
        } else if (mn >= d) {
            std::cout << i + 1 << "-\n";
        }
    }
}

int main() {
    std::ios_base::sync_with_stdio(false);
    std::cin.tie(0);
    std::cout.tie(0);
    solve();
    return 0;
}
```

Для эффективного решения необходимо поддерживать «минимум и максимум в окне» из последних m элементов. Этот известный алгоритм можно реализовать с помощью структуры данных дек или стек. Более подробно о задаче и ее решениях можно прочитать, например, по ссылке: <https://codeforces.com/blog/entry/71687?locale=ru>. Решение задачи с использованием дека на языке Python:

```
from sys import stdin
from collections import deque

inp = stdin.readline

def solve():
    n, m, u, d = int(inp()), int(inp()), int(inp()), int(inp())

    inf = 10**9
    mn = deque([])
    mx = deque([])
    for i in range(n):
        x = int(inp())
        while len(mn) > 0 and mn[0][1] + m - 1 < i:
            mn.popleft()
        while len(mx) > 0 and mx[0][1] + m - 1 < i:
            mx.popleft()

        buy = (len(mn) > 0 and mn[0][0] + u <= x)
        sell = (len(mx) > 0 and mx[0][0] - d >= x)
        if buy and sell:
            print(i + 1, '?')
        elif buy:
            print(i + 1, '+')
        elif sell:
```

```
print(i + 1, '-')

while len(mn) > 0 and mn[-1][0] > x:
    mn.pop()
mn.append((x, i))
while len(mx) > 0 and mx[-1][0] < x:
    mx.pop()
mx.append((x, i))

solve()
```