Задача A. Present Cubinuous

На длину коробки есть 2 ограничения:

- Возможность разместить все квадраты 2×2.
- Суммарная площадь всех квадратов должна быть не меньше площади коробки.

Обозначим длину коробки L. Тогда каждое из этих условий даёт нам минимальную длину, которую должна иметь коробка:

- Если ширина коробки равна K, то по вертикали можно поместить $\lfloor \frac{K}{2} \rfloor$ квадратов (деление с округлением вниз, на Python обозначается //, а в C++ используется /). Значит, чтобы поместить все квадраты, длина коробки $L \geqslant 2 \cdot \lceil \frac{A}{K//2} \rceil$ (округление вверх).
- Суммарная площадь всех квадратов должна быть не меньше площади коробки. Значит, длина коробки $L \geqslant \lceil \frac{4 \cdot A + B}{k} \rceil$.

Значит, чтобы оба условия выполнялись, длина коробки должна быть равна максимуму из $2 \cdot \lceil \frac{A}{K//2} \rceil$ и $\lceil \frac{4 \cdot A + B}{k} \rceil$.

Чтобы вычислить $\lceil \frac{a}{b} \rceil$, можно использовать функцию *ceil*, которая округляет вверх число с плавающей точкой, и ответом будет $math.ceil(a/b)$. Во избежание использования нецелых чисел можно вычислить деление с округлением вверх в целых числах: $(a + b - 1)//b$.

Итоговый код решения выглядит так:

```
a = int(input())
b = int(input())
k = int(input())

squares_vertical = k // 2
cond1 = 2 * ((a + (squares_vertical - 1)) // squares_vertical)
cond2 = (4 * a + b + (k - 1)) // k
print(max(cond1, cond2))
```

Задача B. Акция на день рождения

Обозначим за x количество кексов, которое должна купить Василиса, а за y — количество печений. Тогда нужно решить систему:

$$\begin{cases} x + y = N \\ A \cdot x + B \cdot y = K \end{cases}$$

Вычитая первую строку из второй A раз, получаем

$$\begin{cases} x + y = N \\ (B - A) \cdot y = K - N \cdot A \end{cases}$$

Отсюда, $y = \frac{K}{B-A}$. Причём, если это число нецелое, или меньше 0, или больше N, то ответом будет «-1». Иначе осталось найти x: он равен $N - y$.

Итоговый код решения выглядит так:

```
K = int(input())
N = int(input())
A = int(input())
B = int(input())

y = (K - A * N) // (B - A)
```

```
if y < 0 or y > N or (K - A * N) % (B - A) != 0:
    print(-1)
else:
    x = N - y
    print(x, y)
```

Задача C. Змейка и яблоки

Для начала пронумеруем клетки от 0 до $n \cdot m - 1$ по строкам. Изначально голова змейки находится в клетке 0. За одну секунду голова змейки будет перемещаться из клетки i в клетку $i + 1$, а если $i = n \cdot m - 1$, то змейка достигла конца поля, и голова телепортируется в клетку 0.

Будем поддерживать текущее положение головы *head* и хвоста *tail* (последней клетки змейки). Тогда при появлении каждого яблока нужно сделать следующее:

- Проверить, появилось ли оно внутри тела змейки. Есть 2 случая расположения головы и хвоста:

  1) $head \geqslant tail$, то есть голова не перешла через конец поля. Тогда яблоко появилось внутри змейки, если $tail \leqslant apple \leqslant head$.

  2) $head < tail$, то есть голова перешла через конец поля, а хвост ещё нет. Тогда яблоко находится внутри змейки, если $apple \leqslant head$ или $apple \geqslant tail$.

  В этом случае змейка не будет двигаться.

- Если яблоко появилось не внутри змейки, то нужно посчитать, сколько секунд змейка будет двигаться до яблока. Время движения t равно $apple - head$ по модулю $n \cdot m$, то есть если $apple \geqslant head$, то змейка сделает $apple - head$ шагов, а если $apple < head$ (то есть змейке нужно перейти через конец поля, чтобы доползти до яблока), то она сделает $n \cdot m - head + apple$ шагов.

  В итоге нужно прибавить t к ответу, а также подвинуть голову на t (так как она должна оказаться в клетке с яблоком) и хвост на $t - 1$ (так как длина змейки увеличится на 1).

Код решения:

```
n, m, q = map(int, input().split())
size = n * m
head = 0
tail = 0
ans = 0
for i in range(q):
    row, column = map(int, input().split())
    apple = (row - 1) * m + column - 1
    if tail <= apple <= head or apple <= head < tail or head < tail <= apple:
        continue
    time = (apple - head) % size
    ans += time
    head = apple
    tail = (tail + time - 1) % size
print(ans)
```

Задача D. Многочисленные монеты

Решение на 40 баллов: подсчитаем стоимость одной монеты типа i в монетах типа 1 для всех i. Будем поддерживать стоимость набора в монетах 1 типа. Тогда обновлением будет добавление стоимости добавленных монет к сумме, а для ответа на запрос нужно сравнить стоимость набора и стоимость монет. Длинная арифметика в Python позволит работать с достаточно большими числами, но этого решения достаточно для прохождения только части тестов.

```
n = int(input())
a = list(map(int, input().split()))
val = [1]
for i in a:
    val.append(val[-1] * i)

b = list(map(int, input().split()))
sm = 0
for i in range(n):
    sm += b[i] * val[i]

q = int(input())
ans = []
for _ in range(q):
    t, x, y = map(int, input().split())
    if t == 1:
        sm += x * val[y - 1]
    else:
        print(int(sm >= x * val[y - 1]))
```

Для полного решения нужно заметить, что стоимость монет возрастает логарифмически. В частности, 1 монета типа i стоит хотя бы 2^{60} монет типа $i - 60$. Это значит, что для нахождения ответа не нужно проверять слишком много разных типов монет.

Итого, для решения задачи можно сделать следующее: изначально и при каждом прибавлении монет будем жадно обменивать монеты на более дорогие, пока это можно сделать. Так, для каждого типа монет i (кроме последнего), их количество не будет превосходить a_i. Каждое обновление будет происходить амортизированно за $O(log(C))$. Для ответа на 2 тип запросов можно сделать следующее: если монета максимального типа в наборе имеет тип хотя бы на 60 больше, чем монета в запросе, то только одна эта монета стоит больше, чем монеты в запросе, поэтому ответ «1». Иначе нам нужно рассмотреть не более 60 типов монет.

Код решения:

```
n = int(input())
a = list(map(int, input().split())) + [0]
b = list(map(int, input().split()))
max_nominal = -1
for i in range(n):
    if i < n - 1:
        b[i + 1] += b[i] // a[i]
        b[i] %= a[i]
    if b[i] and i > max_nominal:
        max_nominal = i
q = int(input())
for _ in range(q):
    t, x, y = map(int, input().split())
    y -= 1
    if t == 1:
        i = y
        b[i] += x
        if i > max_nominal:
            max_nominal = i
        while i < n - 1 and b[i] >= a[i]:
            b[i + 1] += b[i] // a[i]
```

```
            b[i] %= a[i]
            if i + 1 > max_nominal:
                max_nominal = i + 1
            i += 1
    else:
        if max_nominal >= y + 60:
            print(1)
        elif max_nominal < y:
            print(0)
        else:
            s = 0
            for i in range(max_nominal, y - 1, -1):
                s = s * a[i] + b[i]
            if s >= x:
                print(1)
            else:
                print(0)
```