**Решения задач**

1. ***Каркас (100 баллов)***

В этой задаче необходимо вывести закономерность (формулу), по которой суммарная длина балок зависит от количества рёбер.

*N*

*M*

*K*

Имеются три вида рёбер в зависимости от их направления — в длину, в ширину и в высоту склада.

Подсчитаем количество рёбер, направленных в длину склада. В длину умещается *N* таких рёбер. В ширину выходит (*M* + 1) таких линий, а в высоту — (*K* + 1) таких слоёв. Общее количество рёбер такого типа — *N* · (*M* + 1) · (*K* + 1).

Аналогично подсчитаем количество рёбер, направленных в ширину склада, — *M* · (*N* + 1) · (*K* + 1).

Количество рёбер, направленных в высоту склада, — *K* · (*N* + 1) · (*M* + 1).

Сложим количества рёбер всех трёх видов.

Итоговая формула — .

Другой вид формулы —.

Пример решения задачи на Pascal:

 var

 N, M, K: Integer;

begin

 Read(N, M, K);

 Writeln(N\*(M+1)\*(K+1)+M\*(N+1)\*(K+1)+K\*(M+1)\*(N+1));

end.

Пример решения задачи на Python:

n, m, k = map(int, input().split())

print(n\*(m+1)\*(k+1)+m\*(n+1)\*(k+1)+k\*(m+1)\*(n+1))

1. ***Подстанция*** (100 баллов)

При любом расположении подстанции «скорой помощи» наиболее удалённым от неё населённым пунктом является населённый пункт либо с наименьшим, либо с наибольшим номером километра. Поэтому для решения задачи достаточно найти наибольший и наименьший номера, а затем путём перебора номеров определить номер, для которого максимум расстояний до этих двух номеров минимален. При равенстве расстояний предпочтение отдаётся меньшему номеру.

Можно выполнить предварительную сортировку номеров населённых пунктов. Это облегчит сразу поиск минимального и максимального номеров, а также выбор нужного населённого пункта при переборе.

Пример решения задачи на Pascal:

 var

 N, i, MaxA, MinA, M, R, D: Integer;

 A: array [1 .. 100] of Integer;

begin

 Readln(N);

 MaxA := 0;

 MinA := 101;

 for i := 1 to N do

 begin

 Read(A[i]);

 if A[i] > MaxA then

 MaxA := A[i];

 if A[i] < MinA then

 MinA := A[i];

 end;

 R := 101;

 M := 101;

 for i := 1 to N do

 begin

 if A[i] - MinA > MaxA - A[i] then

 D := A[i] - MinA

 else

 D := MaxA - A[i];

 if (D < M) or (D = M) and (A[i] < R) then

 begin

 M := D;

 R := A[i];

 end;

 end;

 Writeln(R, ' ', M);

end.

Пример решения задачи на Python:

input()

a = list(map(int, input().split()))

a.sort()

pm = 0

d = 100

for p in a:

 if max(p - a[0], a[-1] - p) < d:

 pm = p

 d = max(p - a[0], a[-1] - p)

print(pm, d)

1. ***Гирлянда*** (100 баллов)

Пусть дано *a* красных, *b* зелёных и *с* синих лампочек. Требуется определить количество перестановок из *a*+*b*+*c* лампочек с учётом того, что лампочки с одинаковым цветом не различаются.

Если бы все лампочки имели разные цвета, то количество всех возможных перестановок было бы равно факториалу их общего количества: . Но поскольку в каждой перестановке есть одинаковые лампочки в одном и том же количестве, некоторые группы перестановок будут неразличимы. Так, например, перестановки, различающиеся только порядком красных лампочек, будут на вид одинаковы, и существует способов переставить красные лампочки без изменения вида гирлянды. При этом в каждом случае независимо от красных лампочек можно переставлять зелёные и синие. Таким образом, в общем числе перестановок, равном , каждая действительно уникальная перестановка будет повторяться раз. Получаем формулу для количества различных способов включения гирлянды: .

Поскольку максимальное общее число лампочек  — 20, а факториал этого числа не превышает 1019, для решения задачи достаточно 64-разрядных целочисленных типов данных.

Функция факториала может присутствовать в стандартных библиотеках языка программирования или может быть написана самостоятельно.

Пример решения задачи на Pascal:

 var

 a, b, c: Integer;

 function f(n: Integer): Int64;

 var

 i: Integer;

 r: Int64;

 begin

 r := 1;

 for i := 1 to n do

 r := r \* i;

 f := r;

 end;

begin

 Read(a, b, c);

 Write(f(a + b + c) div f(a) div f(b) div f(c));

end.

Пример решения задачи на Python:

def f(n):

 return 1 if n == 0 else n \* f(n - 1)

a, b, c = map(int, input().split())

print(f(a + b + c) // f(a) // f(b) // f(c))

Пример решения задачи на Python (использована библиотечная функция):

from math import factorial as f

a, b, c = map(int, input().split())

print(f(a + b + c) // f(a) // f(b) // f(c))

1. ***Кунг-фу*** (100 баллов)

Один из способов решения задачи — моделирование: создать массив из числа элементов, равного числу учеников, и совершать «отборы» на обучение, проходя по массиву с соответствующим шагом и увеличивая значения элементов на 1. В итоге количество элементов массива, равных числу учителей, и будет равным максимально возможному количеству мастеров. Однако при заданных ограничениях на входные данные это решение неэффективно по времени, а по памяти практически нереализуемо на больших входных данных.

Идея эффективного решения: ученик будет изучать *i*-й стиль кунг-фу, если его порядковый номер делится на *ai*. Ученик будет изучать все стили и может стать Мастером, если его порядковый номер делится на все значения *a*1, *a*2, …, *aK*. Минимальный из таких номеров *t* равен наименьшему общему кратному (НОК) всех значений *a*1, *a*2, …, *aK*. Тогда следующие потенциальные Мастера будут иметь порядковые номера, кратные *t*, то есть 2*t*, 3*t*, … Для определения их количества нужно разделить общее количество учеников на *t* нацело.

Для вычисления НОК можно использовать встроенные функции языка программирования или следующие соотношения:

НОД двух чисел может быть вычислен с использованием встроенной функции или, например, с помощью алгоритма Евклида.

Задача может быть решена в пределах 64-разрядного целочисленного типа данных, если выполнить проверку того, что значение *t* не превышает общего количества учеников.

Пример решения задачи на Pascal:

 var

 n, t: Int64;

 k, i: Integer;

 a: array [1 .. 20] of Integer;

 function gcd(a, b: Int64): Int64;

 begin

 if (a = 0) or (b = 0) then

 gcd := a + b

 else

 gcd := gcd(b, a mod b);

 end;

 function lcm(a, b: Int64): Int64;

 begin

 lcm := a div gcd(a, b) \* b;

 end;

begin

 Read(n, k);

 t := 1;

 for i := 1 to k do

 begin

 Read(a[i]);

 if t <= n then

 t := lcm(t, a[i]);

 end;

 if t <= n then

 Write(n div t)

 else

 Write(0)

end.

Пример решения задачи на Python:

def gcd(a, b):

 while a > 0 and b > 0:

 if a > b:

 a = a % b

 else:

 b = b % a

 return a + b

def lcm(a, b):

 return a \* b // gcd(a, b)

n, k = map(int, input().split())

a = list(map(int, input().split()))

t = 1

for x in a:

 t = lcm(t, x)

print(n // t)

Пример решения задачи на Python (использована библиотечная функция):

from math import lcm

n, k = map(int, input().split())

a = list(map(int, input().split()))

print(n // lcm(\*a))