A00. Оленька и Дед Мороз

Первый вариант.
Для решения необходимо найти сумму произведений переменных на соответствующее каждой из них количество начисляемых баллов.

```
# решение участника

a, b, c, d, e = map(int,input().split())
print(a * 5 + b * 4 + c * 3 + d * 2 + e)
```

Второй вариант.
Решение в одну строку.

```
print((lambda: sum(el * i for el, i in zip(map(int, input().split()), range(5, 0, -1))))())
```

A01. Периметр (v.3.0)

Первый вариант.
Для упрощения рассмотрим горизонтальные и вертикальные отрезки, составляющие границу фигуры, отдельно. Сумма всех горизонтальных отрезков равна `w * 2`.
Вертикальные отрезки можно разбить на две части: “внешнюю” и получаемую за счет “внутренних вырезов”. “Внешняя” часть дает `h * 2`, “внутренние вырезы” дают `2 * (h - 1)`.
Осталось подсчитать количество “внутренних вырезов”, зависящее от ширины фигуры: `(w - 1) // 2`.

```
h, w = map(int, input().split())
print(w * 2 + h * 2 + (w - 1) // 2 * 2 * (h - 1))
```

Второй вариант.
Решение с помощью цикла, в котором перебираются индексы столбиков.

```
# решение участника

h, w = map(int,input().split())
cnt = 2
for i in range(w):
   if i % 2 == 0:
       cnt += h * 2
   else:
       cnt += 2
print(cnt)
```

A02. Два квадрата

Первый вариант.
Первый квадрат наибольшей площади из прямоугольника с различными смежными сторонами получается при стороне квадрата, равной наименьшей из смежных сторон, второй квадрат получается со стороной, равной минимуму из двух значений: наименьшей из смежных сторон и разностью между наибольшей и наименьшей сторонами (у нас исходный прямоугольник “уменьшился” по большей стороне на величину наименьшей стороны). Необходимо учесть два особых случая:

1. Если изначально прямоугольник представляет собой квадрат со стороной в одну клетку. В этом случае из него нельзя вырезать два квадрата.
2. Если изначально прямоугольник представляет собой квадрат со стороной более одной клетки. В этом случае из него первым вырезается квадрат со стороной, меньшей исходной стороны квадрата на 1 клетку. Второй квадрат получается размером 1*1.

```
n, m = map(int, input().split())
if n == m == 1:
   print(0)
elif n == m:
   print((n - 1) ** 2 + 1)
else:
   print(min(n, m) ** 2 + min(min(n, m), max(n, m) - min(n, m)) ** 2)
```

```
# решение участника

a, b = map(int,input().split())
if a < 2 and b < 2:
   print(0)
elif a == b:
   k = (a - 1) ** 2 + 1
   print(k)
elif a >= 2 or b >= 2:
   c = max(a, b)
   d = min(a, b)
   e = d ** 2
   f = c - d
   if f >= d:
       g = e * 2
   else:
       g = e + f ** 2
   print(g)
```

A03. Дистанция

Первый вариант.
Решение сводится к нахождению x из уравнения x / N - x / K = S, откуда x = S * N * K / (N - K).
От дробной части избавляемся с помощью целочисленного деления.

```
n, k, s = map(int, input().split())
print(s * n * k // (n - k))
```

A04. Драмкружок, кружок по фото

Первый вариант.
Задача относится к задачам по дискретной математике.
Первоначально находим общее количество записавшихся (n - k), затем записавшихся одновременно в оба кружка (пересечение), для чего из суммы количества записавшихся в каждый из кружков (d + v) вычтем общее количество записавшихся (d + v - (n - k)). Для получения окончательных ответов вычитаем полученное пересечение из записавшихся в каждый из кружков.

```
n, k, d, v = map(int, input().split())
#print(d - (d + v - (n - k)), v - (d + v - (n - k)))
print(n - k - v, n - k - d)


# решение участника

vsego, net, dzudo, voley = map(int, input().split())
dety = vsego - net
a = dzudo + voley - dety
print(dzudo - a, end = " ")
print(voley - a)
```

A05. Подбор пароля

Первый вариант.
Задача относится к задачам по комбинаторике.
Если строка начинается со звездочки, то на этом месте может находиться любой из 26 символов (только буквы), если строка заканчивается звездочкой, то этом месте может находиться любой из 10 символов (только цифры), если звездочка имеется внутри строки, то на ее месте может находиться любой из 36 символов (буквы и цифры). Ответ получается как произведение полученных значений, получаемых для каждой звездочки.

```
a = input()
res = 1
for i in range(len(a)):
   if a[i] == '*':
       if i == 0:
           res *= 26
       elif i == len(a) - 1:
           res *= 10
       else:
           res *= 36

print(res)


# решение участника

a = input()
c = 36 ** a[1:-1].count("*")
if a[0] == "*":
   c *= 26
if a[-1] == "*":
   c *= 10
print(c)
```

A06. Овёс нынче дорог

Первый вариант.
Если рассматривать расстояния до места встречи в виде отрезков, то задача сводится к нахождению минимума удвоенных сумм трех отрезков до места встречи (от трех деревень до места встречи). Сумма будет минимальной, если местом встречи будет любая из деревень, расположенных “внутри”, есть не являющаяся “крайней”. Предварительно требуется расположить деревни упорядоченно, отсортировав заданные координаты.

```
*a, = map(int, input().split())
a.sort()
print((a[3] - a[0] + a[2] - a[1]) * 2)
```

Второй вариант.
Можно рассмотреть все четыре случая, назначая каждую из деревень в качестве места встречи, и найти минимальное значение среди рассмотренных случаев.

```
# решение участника

coords = [int(i) for i in input().split()]
print(min(sum(abs(house - place) * 2 for house in coords) for place in coords))


# решение участника

s = [int(i) for i in input().split()]
ans = float('inf')
for i in range(4):
    v = 0
    for j in range(4):
        v += abs(s[j] - s[i]) * 2
    if v < ans:
        ans = v
print(ans)
```

A07. Шарики и пирамида (v.4.0)

Первый вариант.
Задача решается с помощью цикла. Перебирая уровни, подсчитываем количество шариков на каждом уровне и уменьшаем количество остающихся шариков, пока оно не станет отрицательным, после чего корректируем ответ.

```
n = int(input())
s = lev = cnt = 0
while s <= n:
   cnt += 1
   lev += cnt
   s += lev
print(cnt - 1)
```

```
# решение участника

n = int(input())
v, i, l = 1, 1, 1
while v <= n:
   v, i, l = v+l+i+1, i+1, l+i+1
print(i-1)
```

A08. Комментатор (v.2.0)

Первый вариант.
Задача решается с помощью методов строк. Первоначально удаляются вне нераспознанные буквы, затем в цикле сравниваются соседние (смежные) буквы строки, по их соотношению формируется буква, добавляемая к ответу.

```
a = input().replace('?', '')

res = pr = ''
for el in a:
   if res == '':
       res = el
       continue
   if el.lower() == res.lower():
       if el == res.upper():
           res = el
       continue
   el = el.lower()
   if el != pr:
       res += el
   pr = el

if res:
   print(res)
```

Второй вариант.
Сравнение происходит между проверяемой буквой и последней буквой формируемого ответа.

```
# решение участника

s1 = input()
s = ''
for el in s1:
   if el != '?':
       s += el
if s == '':
   exit(0)
ans = s[0].lower()
for el in s:
   if el.lower() != ans[-1] and (len(ans) != 1 or el.upper() != ans):
       ans += el.lower()
   if len(ans) == 1 and el.lower() == ans[-1] and el.upper() == el:
       ans = el
print(ans)
```

A09. Последняя цифра (v.2.0)*

Первый вариант.
Поскольку входные данные не позволяют решить задачу на Python с помощью прямого возведения в степень, необходимо определить закономерность и увидеть, что последние цифры степеней повторяются с определенным шагом (для разных цифр шаг разный, но он не превышает 4). Поэтому можно оперировать остатком от деления основания степени на 10 и остатком от деления показателя степени на 4. Особый случай - возведение в степень 0.

```
n, k = map(int, input().split())
print((n % 10) ** (k % 4 + 4 * (k % 4 == 0)) % 10 if k else 1)


# решение участника

N, K = map(int, input().split())
print(((N % 10) ** min(K % 4 + 4, K)) % 10)
```

Второй вариант.
Решение с использованием стандартной функции pow() и компилятора PyPy.

```
# решение участника

n, k = map(int, input().split())
result = pow(n % 10, k, 10)
print(result)
```

Решение в одну строку.

```
print((lambda n, k: (n % 10) ** (k % 4 + 4) % 10 if k else 1)(*map(int, input().split())))
```

A10. Путь короля

Первый вариант.
Задача решается с помощью цикла. Изменяются координаты начальной клетки (каждая в нужную сторону, то есть по направлению к конечной клетке) , пока они не совпадут с координатами конечной клетки.

```
row = '87654321'
col = 'abcdefgh'

a, b = input().split()
res = [a]

i_start, j_start = row.index(a[1]), col.index(a[0])

i_stop, j_stop = row.index(b[1]), col.index(b[0])

while i_start != i_stop or j_start != j_stop:
   i_start += 0 if i_start == i_stop else -1 if i_start > i_stop else 1
   j_start += 0 if j_start == j_stop else -1 if j_start > j_stop else 1

   res += [col[j_start] + row[i_start]]

print(*res, sep='-')


# решение участника

st1, st2 = input().split()
while st1 != st2:
   print(st1 + '-',end='')
   if ord(st1[0]) > ord(st2[0]):
       st1 = chr(ord(st1[0]) - 1) + st1[1]
   elif ord(st1[0]) < ord(st2[0]):
       st1 = chr(ord(st1[0]) + 1) + st1[1]
   if st1[1] > st2[1]:
       st1 = st1[0] + str(int(st1[1]) - 1)
   elif st1[1] < st2[1]:
       st1 = st1[0] + str(int(st1[1]) + 1)
print(st1)
```

A11. Максимальный периметр

Первый вариант.
При решении необходимо учитывать, что по условиям задачи четырехугольник является невырожденным, то есть его длина наибольшей стороны должна быть меньше суммы длин трех других сторон. Задача решается с помощью предварительной сортировки по убыванию имеющихся отрезков, из которых находится максимальная сумма четырех подряд идущих при условии, что эти четыре отрезка образуют невырожденный четырехугольник.

```
n, *a = map(int, input().split())
a.sort(reverse=True)
res = 0
for i in range(n - 3):
   if a[i] < sum(a[i + 1:i + 4]):
       res = sum(a[i:i + 4])
       break

print(res)


# решение участника

a, *l = [int(el) for el in input().split()]
l = sorted(l)
i = a - 1
while i > 2 and l[i] >= l[i - 1] + l[i - 2] + l[i - 3]:
   i -= 1
if i <= 2:
   print(0)
else:
   print(l[i] + l[i - 1] + l[i - 2] + l[i - 3])
```

A12. Прачечная

Первый вариант.
Задача решается с помощью левостороннего двоичного (бинарного) поиска, поскольку требуется наименьшее время. Для каждого проверяемого значения времени определяется количество постиранного белья.

```
k, n, *a = map(int, input().split())

w = a[::2]
d = a[1::2]

l = 0
r = ((k - 1)// w[0] + 1) * d[0] + 1
while l < r - 1:
   m = (l + r) // 2
   s = sum(m // d[i] * w[i] for i in range(n))
   if s >= k:
       r = m
   else:
       l = m

print(r)


# решение участника

k, n, *inp = (int(i) for i in input().split())
n2 = n * 2

lower = 0
upper = sum(-(k // -inp[i]) * inp[i + 1] for i in range(0, n2, 2))

while lower < upper - 1:
   mid = (lower + upper) // 2
   if sum(mid // inp[i + 1] * inp[i] for i in range(0, n2, 2)) >= k:
       upper = mid
   else:
       lower = mid

print(upper)
```

A13. Пятница, 13

Первый вариант.
Задача решается с помощью модуля datetime. С помощью его методов устанавливаем в качестве проверяемого заданный год, перебираем все 12 месяцев в нем и определяем, сколько раз в них 13-е число приходится на пятницу.

```
import datetime

a = int(input())
s = 0
for i in range(1, 13):
    if datetime.date.isoweekday(datetime.date(a, i, 13)) == 5:
        s += 1

print(s)


# решение участника

import calendar

n = int(input())
c = 0
for i in range(1, 13):
    if calendar.weekday(n, i, 13) == 4:
        c += 1
print(c)
```

Второй вариант.
Задача решается с помощью определения дня недели, с которого начинается проверяемый год.В невисокосном году 365 дней, в високосном - 366. Таким образом, если невисокосный год начинается с пятницы, то он пятницей и завершается, если невисокосный год начинается с четверга, то он заканчивается пятницей, начинается с пятницы - заканчивается субботой. Найдем, с какого дня недели начинается проверяемый год. При переходе от текущего года к последующему (если он больше текущего) день недели, с какого начинается последующий год, переходит на следующий день недели (например, с понедельника на вторник), если текущий год невисокосный, или на два дня недели вперед, если текущий год високосный (например, с понедельника на среду). Аналогично при переходе от текущего года к предыдущему, но только переход дне происходит в обратную сторону., Отталкиваясь от информации за 2024 год, перебирая все года до проверяемого, можно получить информацию о том, с какого дня недели начинается проверяемый год, а затем, перебирая месяцы в проверяемом году, определить количество 13-х дней каждого месяца, приходящихся на пятницу.

```
def leap(year):
    if year % 4 == 0 != year % 100 or year % 400 == 0:
        return True
    return False


n = int(input())

start = 0
if n > 2024:
    for i in range(2024, n):
        start += 1 + leap(i)
elif n < 2024:
    for i in range(n, 2024):
        start -= 1 + leap(i)

start %= 7

x = [31, 28 + leap(n), 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

res = 0
for i in range(12):
    if (sum(x[:i]) + 13 + start) % 7 == 5:
        res += 1

print(res)
```

A14. Светофорная гирлянда

Первый вариант.
Задача решается с помощью динамического программирования. Первоначально ищется ответ на вопрос, сколько вариантов гирлянды не содержит “светофорную” последовательность, а затем из общего числа вариантов гирлянды вычитается найденное число.

```
import sys
sys.set_int_max_str_digits(5000)

n = int(input())

s = 0
a = [0] * (n + 1)
b = [0] * (n + 1)
c = [0] * (n + 1)
a[1] = b[1] = c[1] = 1
for i in range(2, n + 1):
   b[i] = b[i - 1] + a[i - 1] + c[i - 1]
   a[i] = a[i - 1] + c[i - 1] + b[i - 1] - c[i - 2]
   c[i] = a[i - 1] + c[i - 1] + b[i - 1] - a[i - 2]

print(3 ** n - (a[n] + b[n] + c[n]))



# решение участника (PyPy)

n = int(input())
# зж, кж, только з, только к, только ж, done
dp = [[0, 0, 1, 1, 1, 0]]
for i in range(1, n):
   s = dp[i - 1]
   dp.append([s[2], s[3], s[0] + s[2] + s[3] + s[4], s[1] + s[2] + s[3] + s[4], s[0] + s[1] +
s[4], s[0] + s[1] + s[5] * 3])
print(dp[-1][5])
```

A15. Легендарная матрица (v.15.0)

Первый вариант.
Задача решается с помощью генератора списков.

```
x = [[1, 2, 3], [8, 9, 4], [7, 6, 5]], [[1, 8, 7], [2, 9, 6], [3, 4, 5]]

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

s = [[x[(i + j) % 2][z] for j in range(k // 3)] for i in range(n // 3) for z in range(3)]

for el in s:
   print(*sum(el, []))
```

Другие варианты.

```
# решение участника

n, k = map(int, input().split())
lst = ["1 2 3 ", "8 9 4 ", "7 6 5 ", "1 8 7 ", "2 9 6 ", "3 4 5 "]
for i in range(n):
   for j in range(k // 3):
       print(lst[(i + j * 3) % 6], end = "")
   print()


# решение участника

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

for i in range(a//3):
   c1 = ''
   c2 = ''
   c3 = ''
   for j in range(b//3):
       if (j + i) % 2 == 0:
           c1 += '1 2 3 '
           c2 += '8 9 4 '
           c3 += '7 6 5 '
       else:
           c1 += '1 8 7 '
           c2 += '2 9 6 '
           c3 += '3 4 5 '
   print(c1)
   print(c2)
   print(c3)


# решение участника

n, m = map(int, input().split())
a1 = [[1, 2, 3], [8, 9, 4], [7, 6, 5]]
a2 = [[1, 8, 7], [2, 9, 6], [3, 4, 5]]
for i in range(n):
   for j in range(m // 3):
       if (i // 3) % 2 == 0:
           if j % 2 == 0:
               print(*a1[i % 3], end = ' ')
           else:
               print(*a2[i % 3], end = ' ')
       else:
           if j % 2 == 1:
               print(*a1[i % 3], end = ' ')
           else:
               print(*a2[i % 3], end = ' ')
   print()
```